1987

Using Computer-Aided Software Engineering (CASE)--tools to document the current logical model of a system for DoD requirements specifications.

Ganzer, Donna A.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USING COMPUTER-AIDED SOFTWARE
ENGINEERING (CASE) TOOLS
TO DOCUMENT THE CURRENT LOGICAL MODEL OF
A SYSTEM FOR
DOD REQUIREMENTS SPECIFICATIONS

by

Donna A. Ganzer

September 1987

Thesis Advisor Barry A. Frew

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; Distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) Code 54	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Using Computer-Aided Software Engineering (CASE) Tools to Document the Current Logical Model of a System for DoD Requirements Specifications (u)

12 PERSONAL AUTHOR(S)
Ganzer, Donna A.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1987 September	69

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computer-Aided Software Engineering Tools, CASE Tools, Software Engineering, Abbreviated Systems Decision Paper (ASDP), Structured Analysis and Design

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The Naval Postgraduate School's final exam scheduling system serves as a test case with which to compare two commercially available Computer-Aided Software Engineering (CASE) tools. The tools, Nastec Corporation's *DesignAid* (Release 3.55) and Index Technology's *Excelerator* (Release 1.7), are used to create Section 4.1 of two Abbreviated Systems Decision Papers to determine if their output can satisfy and should replace some of the Life Cycle Management documentation requirements of the Department of Defense for the procurement of Information Systems having a total cost of $100,000 or less.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Barry A. Frew	22b TELEPHONE (Include Area Code) (408) 646-2924 22c OFFICE SYMBOL Code 54Fw

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited.

Using Computer-Aided Software Engineering (CASE) Tools to Document the Current Logical Model of a System for DoD Requirements Specifications

by

Donna A. Ganzer
Captain, United States Marine Corps
B.S., University of Texas at Austin, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1987

ABSTRACT

The Naval Postgraduate School's final exam scheduling system serves as a test case with which to compare two commercially available Computer-Aided Software Engineering (CASE) tools. The tools, Nastec Corporation's *DesignAid* (Release 3.55) and Index Technology's *Excelerator* (Release 1.7), are used to create Section 4.1 of two Abbreviated Systems Decision Papers to determine if their output can satisfy and should replace some of the Life Cycle Management documentation requirements of the Department of Defense for the procurement of Information Systems having a total cost of $100,000 or less.

TABLE OF CONTENTS

LIST OF TABLES

I. INTRODUCTION

A. BACKGROUND

As varied and complex as today's military information needs are and will continue to be, ongoing improvements and refinements in the accuracy and effectiveness of military information systems are crucial to ensure our survival as a nation. Designing the software for these information systems is a labor-intensive. mistake-prone process. The complexity of computer software being designed and contemplated for future military use makes it virtually impossible to expect that manual, paper-and-pencil methods of software design will be adequate to ensure quality products in a timely manner.

What would certainly be welcomed by software designers is a better tool or system of tools to ease the burden of keeping track of the myriad of details involved in software design. This tool should enable the designer to concentrate more on the logic of the design than on its housekeeping. There are such tools on the commercial market--tools which automate to a great extent the detailed checking of design diagrams and the cataloging of information in a centralized data dictionary. These tools have enabled software designers to catch software errors that have plagued programs designed without them, and have significantly increased designer productivity [Ref. 1: p. 234].

The DoD requirements for documentation in the lifecycle management of a system are stultifying. If a tool could be found that would document a system to DoD's satisfaction with less trauma than it is now done, the effects of using such a tool could only be beneficial to the system's design and to its designers.

B. PURPOSE OF THESIS

This thesis uses two commercially-available CASE programs, Nastec Corporation's *DesignAid* and Index Technology Corporation's *Excelerator*, provided by Headquarters, U. S. Marine Corps, to create part of the functional/structural specifications for a program that schedules final exams at the Naval Postgraduate School, a process which is currently performed manually. DoD requirements for documentation of this system's development are accommodated as closely as possible using these CASE tool capabilities. The two CASE programs evaluated as to their

relative merits in the areas of: user-friendliness, flexibility toward user-defined structures, ease with which information can be accessed in program files, quality of design error-checking and ease of correction of errors, quality of graphics/text interfacing, ease of report generation, and suitability of reports generated (from DoD's perspective).

The purpose of this thesis is to determine how effective two particular computer-aided software engineering (CASE) tools are in doing what they claim to do--making system design easier by relieving the burden of keeping track of project details, and to determine whether these CASE tools can satisfy some of the documentation needs of DoD. An assumption is that the two CASE tools evaluated are representative of the tools currently available on the commercial market.

C. RESEARCH QUESTIONS

The primary research questions that are addressed are as follows:

- How easy to use are the CASE tools examined in this thesis?
- Can these CASE tools give DoD a clearer picture of the requirements of a software system than DoD's current specification documents do?
- Could part of the current DoD specification documents be satisfied with output from these CASE tools?

D. OVERVIEW OF METHODOLOGY

A model of the current final exam scheduling system at NPS was used in this thesis as a test case with which to evaluate the capabilities of both *DesignAid* and *Excelerator*. Information about the NPS final exam scheduling system was obtained through interview of the NPS Final Exam Scheduler, Ms. Mary Horn, and through review of the thesis of a former NPS student, Dietmar W. Fiegas.

Once initial familiarity with the CASE products was gained by doing the products' tutorials and reading their documentation, the current logical model of the NPS final exam schedule system was diagrammed and defined by the author using both *DesignAid* and *Excelerator*. Because of the author's familiarity with, and the Marine Corps' usage of the Yourdon methodology of systems analysis and design, it was the methodology that was used to design the data flow diagrams and data dictionary of the scheduling system's current logical model.

The output from these CASE products is used and evaluated as the part of a DoD requirements specification document that examines the current logical model of a system under consideration for improvement or replacement.

No code is generated to actually test the resulting functionality of the designed system.

Specific methodology terminology is contained in Section E of Chapter II of this thesis.

II. OVERVIEW

A. STRUCTURED ANALYSIS AND DESIGN OF SOFTWARE

For years, well-established disciplines such as Engineering have followed a rigid, concise set of rules to design and build systems (bridges, buildings, etc.) formed of amazingly complicated and interwoven components. Every step of the process is closely controlled, coordinated, and documented by the use of some methodology which clearly spells out what output must be obtained from each step before continuing to the next. The system is designed one step at a time, each step completed using the output from the step before it. The result is usually a successful system -- one which has been built methodically and thoroughly, with virtually all of the factors which account for its success being considered during its actual design and construction. The methodology used to create these systems is highly structured in that it requires its steps be accomplished in a specific sequence (however, concurrency between some steps is allowed), and its detailed documentation be approved at each step before proceeding to the next. [Ref. 2: pp. 6 - 8]

The implementation of such a structured methodology has not existed until recently for software engineers. Although structured software methodologies have been discussed in literature and among academics since the early 1970s, they've only been introduced to commercial software organizations since the early 1980s. Obviously, structured software methodologies have a long way to grow to reach the level of maturity that the engineering field's methodologies have enjoyed for at least the last century. To complicate and delay the process of developing a thorough, sound, basic methodology for software development, new products and services appearing almost daily in the computer world are diverting software developers' attention. Products such as powerful personal computers, user-friendly fourth generation programming languages, and application prototyping programs lure developers with claims of being able to make their interaction with users much easier and faster. These products incorporate the latest technology to enable software developers to do easier and faster whatever it was they were doing with the user before the products were invented, but they do not tie together the entire process of analyzing and designing a system from beginning to end. Edward Yourdon, a pioneer in the search for a better way to

develop software, says that new developments in technology are great, but are merely tools to model a structured software development methodology, not replacements for the methodology itself, and that a structured methodology can continue to embrace new technologies as they develop without the underlying concepts of the methodology being destroyed. [Ref. 1: p. 6]

Yourdon has identified seven phases that have traditionally occurred in the creation of a classical (computer) system: [Ref. 3: p. 22]

- Problem Recognition
- Feasibility Study
- Analysis
- Design
- Implementation
- Testing
- Maintenance

He believes that these phases are a valid framework from which to create systems, but that in the past, the lack of a structured methodology to accomplish these phases has hampered software developers and has led to some disappointing results. He has created a structured methodology which gives software developers pictorial (graphic) tools to organize the steps in each of the above phases, analogous to the blueprints and models used by Engineers to build their systems. The tools Yourdon provides seem to be based on the theory that a picture is worth a thousand words; it replaces stacks of written system specifications with pictorial, more easily-comprehended representations of these specifications.

The Feasibility Study, Analysis, and Design phases especially make heavy use of such graphically-oriented tools, since these phases are the ones during which communication between the user(s), analyst(s), and designer(s) happens, and clear descriptions of requirements are essential to make the end product successful. Graphic tools such as Data Flow Diagrams, Data Dictionaries, Process Specifications, Entity-Relationship Diagrams, and State Transition Diagrams help the analyst and user of a developing software system narrow down exactly what the user wants the system to do, and are used mostly during the Feasibility Study and Analysis phases. The analyst then communicates the user's requirements to the designer via tools such as Structure Charts, Module Specifications, and Interface Specifications during the Design phase. [Ref. 3: pp. 39 - 97]

The use of these tools requires just as much attention to detail as the old method of describing what a system should do by writing paragraphs about it. Since they are graphically-oriented, they can be automated and the computer's capability to quickly process large amounts of detail can be used to find errors that would take weeks or months to do manually. Such programs that automate the tools of structured methodology, called Computer-Aided Software Engineering (CASE) tools, are being developed successfully by commercial firms. Not only do these automated tools keep track of the detailed specifications of a system, but they also provide a virtually instant assessment of the impact of requirement changes in the system. Software developers should be relieved of the burden of manual detail-tracking by these tools and be encouraged to concentrate their efforts on making better use of a structured methodology.

B. COMPUTER-AIDED SOFTWARE ENGINEERING (CASE) PRODUCTS

There has been a veritable explosion of automated tool products from 1984 to the present, and it is evident that the few dozen or so products already announced are only the first generation of a family of products that will begin to appear in the marketplace throughout the next ten years. Most of the products introduced so far provide some form of the following critical features:

- *Graphics Support.* Using a hand-held mouse or some other appropriate device, the CASE product allows its user to create a diagram on the screen, and revise it, if necessary, in a matter of minutes.

- *Data Dictionary Support.* Some form of a data dictionary is provided by the CASE product so that the data elements, process names, and other items named in the graphical models are properly defined to the CASE program so that they can be automatically checked for consistency.

- *Consistency Checking.* Probably the most important feature of the CASE product, it ensures that all items named on the graphical models exist in the data dictionary, that items are not defined more than once in the dictionary, and that net inputs and outputs at one level of a data flow diagram correspond exactly to the net inputs and outputs of the parent process in the next higher level diagram.

Yourdon predicts that the use of CASE tools with these and upcoming features will provide a factor-of-ten improvement in software reliability and maintainability: already a 20 to 30 percent increase in productivity has been achieved. [Ref. 1: pp. 234 - 236]

Two of the CASE products currently available are Nastec Corporation's *DesignAid* (Release 3.55) and Index Technology (InTech) Corporation's *Excelerator* (Release 1.77).

1. **DesignAid**

DesignAid is one program in a series of Nastec's CASE 2000 Products which attempts to computerize the process of managing the software development life cycle. It supports the Feasibility Study, Analysis, and Design phases of the lifecycle, as described by Yourdon in section A of this chapter, and supports these phases using Yourdon's terminology and notation. *DesignAid* does not generate code, but it provides an interface to another CASE 2000 program (*Gamma*) which does.

DesignAid uses a data dictionary composed of three files: the definition file, occurrence file, and structure file. The definition file stores the name, type, and other identifying information about an object in a *DesignAid* user's data flow diagram; the occurrence file stores information about what user project file(s) the object resides in; and the structure file stores information about what data elements are contained in the data flows and data stores of the user's project data flow diagrams.

To design a set of data flow diagrams to be analyzed by *DesignAid*, a user completes essentially the following sequence of events:

- Opens a file and puts its file name in a menu file (a menu file is used to keep track of what files have been created by project team members; it serves as a table of contents for the project files. It is MANUALLY created...if a user forgets to put the file name in the menu file, the file will not be readily visible to other team members).
- Draws a context level data flow diagram in this file.
- Enters the objects in the data flow diagram into the definition file of the data dictionary. For each process that explodes to another data flow diagram, the user enters in the process symbol the name of the file to which it explodes before the process is defined to the dictionary.
- Checks the accuracy of the data flow diagram (validates it) to determine any syntax errors in the diagram.
- Enters the objects in the data flow diagram into the occurrence file of the data dictionary.
- Enters the data elements of each data store and data flow into a SEPARATE user-defined project file to be read into the structure file of the data dictionary. The user then enters the file name in the project's menu file.
- Creates a process narrative (mini-specification) file for lowest level processes (a context level diagram will probably have few, if any, of these). The user then enters the process narrative file name in the project's menu file.

- Creates a structure chart file for processes requiring one. The user enters the structure chart file name in the project's menu file.
- Opens more files, draws children data flow diagrams in these files, and updates the definition, occurrence, and structure files of the data dictionary with the appropriate information. The user enters the file names in the project's menu file.
- Balances the parent and children data flow diagrams.

In addition to allowing its user to design data flow diagrams quickly and analyze them for syntactical and balancing errors, *DesignAid* provides other capabilities. The *DesignAid* user can:

- inquire into the contents of the data dictionary although the process is complicated), and audit changes made to the project data dictionary,
- share project data with other users on a network,
- manipulate files from within *DesignAid* with selected DOS commands,
- create macros,
- read files from within other files at a keystroke, and
- access multiple printers.

Nastec Corporation also offers an educational program of professional seminars and workshops in structured analysis and design, project management, and *DesignAid* training, as well as supports a toll-free hotline service. [Refs. 4,5,6]

2. **Excelerator**

Excelerator, like *DesignAid*, supports the Feasibility Study, Analysis, and Design phases of the system lifecycle using other methodologies as well as Yourdon's. The only code that is generated by *Excelerator* transforms screen report formats for the system being designed into files that can later be merged into the code for the entire program (a choice of COBOL, C, BASIC, and PL/1 is available).

Excelerator operates slightly differently than does *DesignAid*. There are five parts to the data dictionary, one for each type of entity (similar to *DesignAid*'s "object") that it stores: data, process, graphs, screens reports, or other. The entity is entered to the data dictionary only once by the user, instead of the three times required by *DesignAid*. The entity is automatically assigned to the appropriate part in the dictionary and *Excelerator* records its location in the project graphs without the user having to invoke *Excelerator* to record it.

To design a set of data flow diagrams to be analyzed by *Excelerator*, a user completes the following steps:

- Opens a graphics file and draws a context level data flow diagram (the file name is automatically stored in the Graphs portion of the data dictionary, and file names are instantly available to the user through simple menu commands).
- Describes the entities to the data dictionary by calling up a description screen, and filling in the appropriate fields.
- Describes a record containing elements of information about data stores and data flows (if the flow is not an element in itself) of the data flow diagram to the dictionary.
- Opens more graphics files and draws children data flow diagrams, linking the parent diagrams to them by filling out the "Explodes to" field in the description screen for the parent process with the name of the child data flow diagram to which it explodes.

Excelerator also draws presentation graphics -- a capability that *DesignAid* does not support. These graphics allow an analyst to sit down with a system's end user at a terminal and draw, in terms familiar to the user, a picture of what the user views his/her system doing now (the current physical model) and what the user wants it to do in the future. Examples of symbols used in a presention graph are shown in Table 1.

The ease with which changes can be made onscreen, and the simplicity and universality of the symbols used, make ideas exchanged between a system's end user and an analyst more tangible and easily understood than they would be if the user and analyst used verbal/textual means, or made time-consuming eraser-and-pencil changes to manually-drawn pictures.

Other features of *Excelerator* include:

- an extensive ability to access the contents of the data dictionary (although access to this information is a complicated process),
- the ability to produce six types of graphs (presentation graphs, data flow diagrams, structure charts, structure diagrams, data model diagrams, and entity-relationship diagrams),
- the ability to design a screen or report to be used by the completed system,
- the ability to share data among multiple users,
- the ability to link to several word processing programs and a project management program, and
- the capability to batch program files to print out in a specified order to comprise a functional specification.

InTech also supports a toll-free hotline service. [Refs. 7,8,9]

TABLE 1

OBJECTS AVAILABLE IN PRESENTATION GRAPHS

C. THE DEPARTMENT OF DEFENSE LIFECYCLE MANAGEMENT PROGRAM

DoD acknowledges the need to manage its information systems' lifetimes from the time the necessity for them is recognized until they no longer satisfactorily perform the function for which they were created. The process of administering a DoD information system throughout the five phases of its development is referred to as the lifecycle management (LCM) of the system [Ref. 10: p. 2].

The objectives of LCM are to:

- identify and assign the responsibilities of various levels of managers throughout the system's lifecyle,
- establish an organizational structure to ensure the effective management of the system under development,
- provide visibility for resource requirements, and
- ensure management accountability for the system's development. [Ref. 11: p. 2]

These objectives are intended to create a continuous span of control over the project's lifetime so that better coordination of limited resources can be effected.

LCM exercises control over the project's lifetime in five chronological phases: [Ref. 10: p. 4]

Mission Analysis/Project Initiation. This phase identifies and validates a need, determines assumptions and constraints on solutions, and makes recommendations for alternative concepts to satisfy the need. The approval of the documentation produced at the completion of this phase, the Mission Element Need Statement (MENS), is necessary for the next phase to occur.

----Milestone I----

Concept Development. The purposes of this phase are to identify user requirements, evaluate alternative methods to satisfy those requirements, and to recommend specific alternatives for further exploration. Approval of the System Decision Paper (SDP), which is the product of this phase, is the catalyst for the next phase.

----Milestone II----

Definition/Design. Functional requirements are fully defined and the information system is technically designed during this phase. Approval of the memorandum which updates the SDP with details of the information system's design is the exit criteria of this phase.

----Milestone III----

System Development. This phase develops, integrates, tests, and evaluates the entire system and is completed when the appropriate manager certifies that the system meets the mission need and approves its implementation.

----Milestone IV----

Deployment and Operation. The purposes of this phase are to implement, operate, and maintain the system. A periodic review is conducted to ensure that the system is still performing up to its functional requirements.

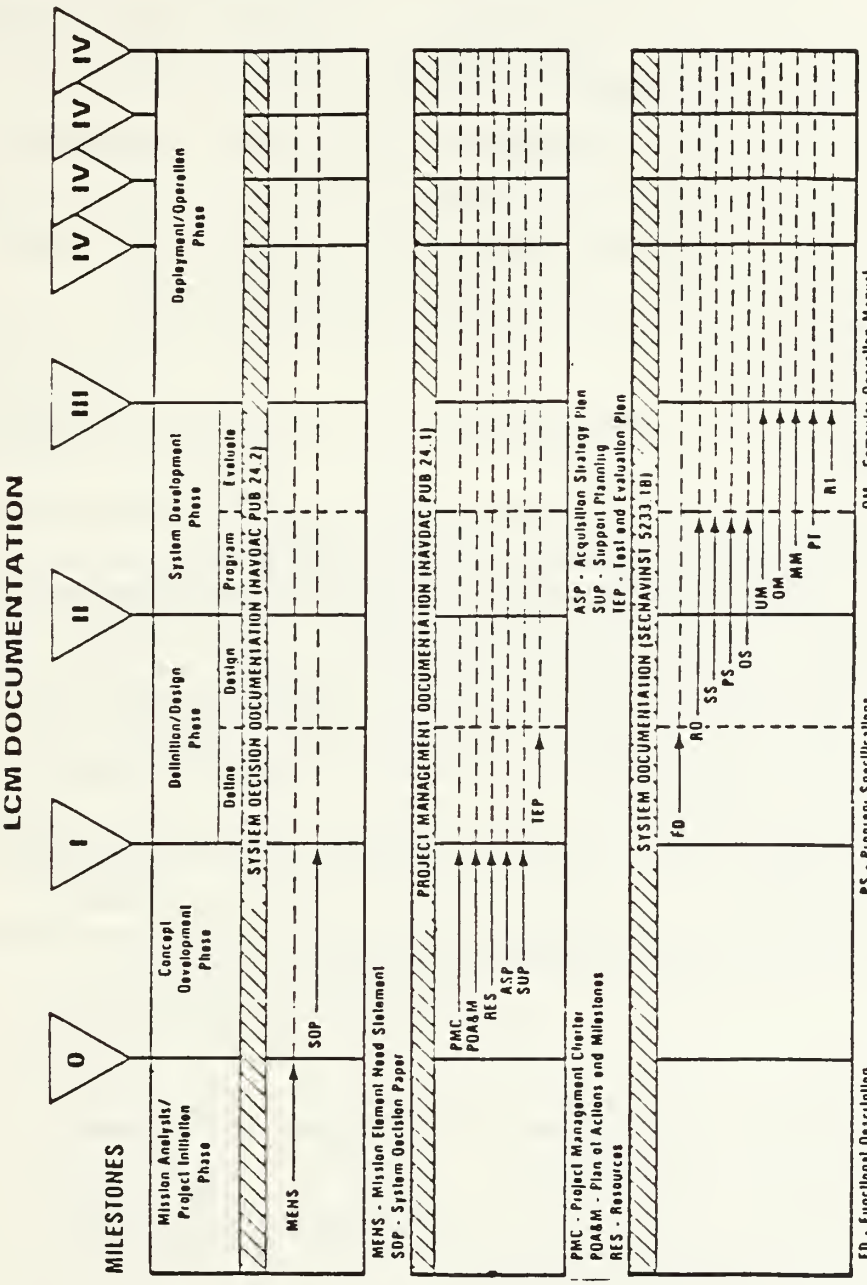
Each of the above phases is separated by a decision point, called a Milestone, at which the decision is made whether to continue on to the next phase or not, based on what was determined during the phase. The phases of the DoD LCM are similar to the ones Yourdon uses in his structured methodology, and the decision documentation required by DoD to proceed from one phase to another is analogous to the tools (data dictionaries, data flow diagrams, etc.) that Yourdon provides to move from one step of his methodology to the next.

Throughout the LCM, three types of documentation of the system's progress are maintained: decision, project, and system documentation. Decision documentation (MENS, SDP, and memoranda updating the SDP) keeps track of the decisions made by appropriate authority at each milestone; project documentation is maintained by the project manager to keep track of the management details of the development of the information system; and system documentation contains detailed functional requirements, design specifications, and technical specifications needed for communication with the people who make the system work. These types of documentation are maintained concurrently, coordinated by the project manager of the system under development. The relationship of these kinds of documentation to the LCM process is illustrated in Table 2 [Ref. 10: encl (4), pp. 13-14].

For small information systems projects (total cost less than $100,000), the decision documentation required for the LCM process can be condensed: an Abbreviated Systems Decision Paper (ASDP) can be produced, which incorporates the essential elements of the MENS and SDP decision documentation of the first and second phases of the DoD LCM process. The approval of the ASDP authorizes the full-scale development of the system without having to produce the decision documentation required between the Definition/Design and Deployment and Operation phases. The format for an ASDP is illustrated in Appendix A. [Ref. 10: p. 5]

TABLE 2

DOD'S LIFE CYCLE DOCUMENTATION TIMETABLE



The Naval Postgraduate School could use an ASDP to request, for example, authorization to implement a small information system which could replace part or all of its final exam scheduling process -- a process which is now done manually.

D. NAVAL POSTGRADUATE SCHOOL FINAL EXAM SCHEDULING PROCESS

Currently, final exams are scheduled manually by two people in the Registrar's office of the Naval Postgraduate School. The process is time-consuming (takes approximately a week), and involves repetitive manual data entry -- a process that seems well suited to being performed by a computer.

Several constraints provide guidelines to the process (Appendix A contains explanations of uppercase terms): [Ref. 12: pp. 24, 33]

- final exams are scheduled within four consecutive days of one week,
- all courses must have a two-hour block of examination time,
- all SEGMENTS of one course must have the exam during the same FINAL EXAM PERIOD,
- every student must have enough space to spread out (at least 1.5 seats per student),
- there must be at most two exams for each student per day and the same SECTION must not have two exams back-to-back,
- faculty members assigned to teach a SEGMENT can only be scheduled to give an exam once per FINAL EXAM PERIOD,
- a course may have more than one ROOM for the exam but all ROOMs for one segment must be on the same floor of one building,
- on request of the instructor, a final exam may be attempted to be scheduled on the first day of finals week,
- ROOMs for the exams should be in designated areas of the campus, as per Table 3, and
- all lecture ROOMs are available for exams during final exams week, except ROOMs occupied by REFRESHER COURSES.

In broad terms, the scheduling process consists of the following steps: [Ref. 13]

1. Assemble the list of courses for which a final exam will be given.
2. Sort the list by size (largest first).
3. Schedule the first course on the list for the first final exam period.
4. Move to the next course on the list.
5. For the current final exam period, do the following:

TABLE 3

COURSES AND THEIR PREFERRED ROOMS ON CAMPUS

COURSE INDICATOR	BUILDING	FLOOR
AE, ME	Halligan	
AS, CO, CM, IS, MN	Ingersol	2, 3
EE	Bullard	
	Spannagel	2, 3, 4
GH	224	
CS, MS	Spannagel	2, 3, 4
MA	Ingersol	2, 3
OA, OS	Ingersol	3
	Root	2, left
MR	Root	2, left
NS. ST, OC	Root	2, right
PH, CC	Spannagel	1, 2

a. Determine which other previously-scheduled course(s) for this final exam period contain at least one of the same sections as this course does. If this course has at least one section in common with a course already scheduled for this exam period, try the next exam period (current final exam period + 1). Keep checking the next final exam period until all twelve periods have been checked, or the constraint is satisfied. If the constraint is satisfied, move to the next step; if not, the course is blocked (see step 6).

b. Determine if this course is a segment of a previously-scheduled course. If so, schedule it at the same time as the previously-scheduled course. If the professor's time schedule does not permit this, reschedule all segments of the course to another exam period, restarting the scheduling process at step 5a. If the segments are not able to be scheduled in any time period, the course(s) is(are) blocked (see step 6).

c. Check for conflicts with the professor's schedule (i.e., he/she is teaching a refresher course or is already scheduled to give an exam during this exam period). If conflicts exist, try the next exam period, restarting the scheduling process at step 5a. If a conflict exists at every time period, the course is blocked (see step 6).

d. Check to see that, by scheduling this course's final for this exam period, the limit of two exams per day per section is not exceeded and no back-to-back finals for the same section are scheduled. If conflict(s) exist, try the next

exam period, restarting the scheduling process at step 5a. If no final exam period is without conflict, the course is blocked (see step 6).

e. Assign an available room according to the room constraints provided. If no room or combination of rooms fulfill the space/location requirements, try the next exam period. If no final exam period is without conflict, the course is blocked (see step 6).

f. If none of steps 5a to 5e have conflicts, record the course, time, and room number on a master schedule. Return to step 4, continuing the scheduling process with the next course.

6. If a course is blocked, and what is causing the blockage is not easily solved (i.e., rescheduling an already-scheduled course to accommodate the blocked course), the entire list of courses may have to be resorted and the scheduling process restarted from scratch.

Obviously, this is a broad-brush view of the scheduling process. Many decision points result from the constraints imposed on the process, and the cumulative effect of these constraints can create conditions that do not allow an exam for a course to be scheduled without many iterations of the process or compromises to it. If, for example, three-quarters of the courses' exams have been scheduled and the next course's exam can't be scheduled in any of the twelve time periods because it fails at least one constraint for each period, a stalemate has occurred in the scheduling process. At this point, a decision must be made whether to reshuffle the list of courses and try the whole process again, hoping that the new order of courses will result in a successful pattern, or to try to determine which course(s) has caused the logjam and reschedule it, which might allow the rest of the scheduling process to continue successfully. [Refs. 12,13: p. 42]

Much of the scheduling process looks like it could be aided by some sort of automation. A software program could possibly be written to lessen the manual handling of part or even all of the final exam scheduling process at the Naval Postgraduate School.

E. SPECIFIC THESIS METHODOLOGY

This thesis uses the Naval Postgraduate School final exam scheduling process as the subject of an Abbreviated System Decision Paper (ASDP), and prepares Section 4.1 of the ASDP using only a Computer-Aided Software Engineering (CASE) tool. Section 4.1 is prepared using each CASE tool evaluated (DesignAid *and* Excelerator), *and are contained in Appendices* C and D. The author describes graphically much of what has previously been described textually in an ASDP, discovering in the process:

- How easy the CASE tools examined in this thesis really are to use,
- how effectively DoD Abbreviated Systems Decision Paper specification requirements for the current logical model of a system (Section 4.1 of ASDP) can be satisfied using CASE tool output, and
- whether Section 4.1 of the ASDP should be replaced by the output of either or both of these CASE tools.

It is assumed that the Naval Postgraduate School has a problem with the current manual scheduling system and can clearly identify the detriment(s) to the school's mission, as required by Section 1 of the ASDP. The Required Capabilities, Cost Analysis, Benefit Analysis, Funding, and Planning Data requirements of Sections 2, 5, 7, and 8 of the ASDP is assumed to be provided by other means. Because this thesis concentrates solely on graphically representing the current logical model of the Naval Postgraduate School final exam scheduling system, no Proposed Alternative required by Section 3 is provided by the author.

An outline of the process by which this thesis was prepared is provided in Section D of Chapter 1 of this thesis.

III. ANALYSIS

It took a relatively long period of time (approximately one month of 8-hour workdays) for the author, an inexperienced system analyst/designer, to become accustomed to the functioning of both *DesignAid* and *Excelerator*. This learning process was performed alone, for the most part, without the aid of formal schooling by either Nastec or InTech representatives. The hotline customer service numbers provided by both companies were of considerable assistance, however, and requests for aid were handled professionally and courteously. The time spent learning the physical details of how each product functions most certainly would have been shortened if a person knowledgeable of the product would have been physically on hand to answer questions as they arose, since much "wheel-spinning" occurred over what turned out to be simple problems whose solutions weren't obvious to the author from reading the product documentation (the hotline was used as a last resort).

Appendix C contains Section 4.1 of an Abbreviated System Decision Paper (ASDP) created from the output of *DesignAid* and Appendix D contains Section 4.1 of an ASDP created from *Excelerator* output. Both appendices are organized generally as follows:

- data flow diagrams in descending order from the context level,
- contents of data stores and data flows,
- description of data elements (because the data elements are self explanatory for the NPS final exam scheduling process, they are described very minimally.).

Both appendices were printed on an ALPS P2000G (Epson FX - compatible) printer, and reduced to fit thesis layout specifications.

A. DESIGNAID

It is not obvious how *DesignAid* collates the information provided to it by its user. The concept of drawing a data flow diagram and then entering information about objects in the diagram into the three distinct files of the program's data dictionary (definition, structure, and occurrence files), so that the data flow diagram can be balanced with its child or parent diagram, is not difficult to understand, but how that concept is implemented inside the data dictionary is not obvious and is not well described by either the program or its documentation. Without this clear picture

of how the data dictionary organizes and manipulates the information it contains, it is difficult for the user to make full sense of the reporting capabilities of *DesignAid*.

1. **Tutorial/User's Guide/Reference Manual**

The Tutorial could have been more clearly written--there are paragraphs that are unclear and some contain different information relative to what actually occurs on the screen in some minor situations. The User's Guide and Reference Manual are broadly comprehensive about describing what *DesignAid* can do, but not in as much detail as is needed. For example, a Reference Manual description of the field "value constraints" in the Object Definition form of the Data Dictionary Menu indicates that the value of an object can be defined to be between two values; that is, the value of an object (presumably a data element) can be in the range of A and Z (denoted (A - Z)), but it is unclear about whether a series of values such as (A, E, J, Z) can be defined. The documentation concentrates more on telling the user what keystrokes to use to enter or manipulate information than it describes how *DesignAid*'s capabilities actually work. This rather one-dimensional approach to writing the user documentation provides an additional hurdle to overcome in learning to use *DesignAid* effectively.

2. **Data Dictionary**

Parent-child diagram balancing problems surfaced during the evaluation of *DesignAid*. Discussion with a technical representative of Nastec Corporation on their customer service hotline revealed a known bug in the *DesignAid* program which prevents it from searching the logical path for a file name of a data flow diagram during the balancing process. Essentially, then, either the absolute path name of the file of the child data flow diagram must be specified in the parent process bubble, or *DesignAid* must be loaded from the directory in which the files reside. [Ref. 14]

This known bug also makes changing anything about an object whose information has already been entered into all files of the data dictionary impossible. According to the User's Guide, to change an item of information about an object, deletion of the object from the dictionary files and then re-entry of the new object information is necessary. Even then, the object can only be deleted if there is no information for it in the occurrence file of the data dictionary. To illustrate, after deciding that a certain process, called "NPS FINAL EXAM SCHEDULING SYSTEM", should be renamed to "FINAL EXAM SCHEDULING SYSTEM" after it was fully defined to the data dictionary files, the author attempted to purge, in order, the occurrence, structure, and definition files for the process (as indicated by the User's

Guide to be the correct procedure for deleting an object from the dictionary so that new information about it could be entered). When deletion from the occurrence file was attempted, a message appeared on the screen, saying, "Occurrence Not Found". When attempts were made to delete the remainder of the information, a message appeared, saying, "Unable to Delete - Occurrences Exist."

An additional difficulty encountered with the data dictionary was in interpreting the data dictionary report output. There is no simple, convenient procedure to "dump the dictionary" to see what the dictionary files contain about the composition of each object in the project, and where it is located in the project. Several different reports--one for each type of object --can be produced and then pieced together to provide such information, but the process is slow and cumbersome. The report generation explanation in the User's Guide and Reference Manual is particularly sketchy, and gives only a broad-brush definition of fields within a report option.

3. **Diagramming**

DesignAid does not create presentation graphs. Presentation graphs are a helpful and necessary tool to focus the attention of the system's end user(s) on the project and to avoid as many current-physical-system misunderstandings as possible. It seems that, to more fully support user/analyst communication, the ability to create presentation graphs should be available.

Some annoyances in *DesignAid*'s diagramming capabilities exist:

- When moving symbols, *DesignAid* allows the moved symbol (and its attached connectors, if any) to be drawn over symbols which aren't being moved and are in the path of the newly-moved symbol. The capability should exist for the moved symbol and its connectors to avoid unmoved symbols.
- When the newly-moved symbols are moved away from the symbols they've overlaid, the symbols which had been overwritten are no longer whole; that is, blank space exists where the trespassing symbol borders overlaid the stationary symbol. There is no refresh capability to make the symbols whole again. Rather, the symbols have to be erased then redrawn.
- Connectors in a data flow diagram cannot be moved by themselves, nor can they be deleted with one command. Deleting a connector involves moving the cursor to mark the beginning of a rectangular area to be deleted, then moving it to the diagonal corner to mark the end of the area to be deleted, then pressing a key to actually delete the area. Moving a connector involves deleting the area it resides in and redrawing it.

Although *DesignAid* can be used with a mouse, the author did not have access to the type of mouse that worked with the software, so commands were issued through

the pull-down menus or through keyboard commands. The use of a mouse would probably make the process of drawing data flow diagrams quicker than the keyboard-entry method.

4. **Validation**

Prior to balancing a parent data flow diagram with its child diagram, validation of the diagram must take place to ensure its symbology and syntax are correct. During validation of the data flow diagrams used in this thesis, error messages appeared which did not describe the actual problem with the diagram. For example, one process name exceeded the maximum allowable character length (50 characters) and received an error message that said that the process below it was not a valid object, and that the data flow line under that process was an invalid symbol. When the process name was changed to one of less than 50 characters, the error messages did not reappear.

During validation of a particular data flow diagram used in this thesis, the data dictionary would not recognize a label on a data flow in the diagram. The label is located within the specified distance it should be from the line, yet a message saying the data flow is unlabeled was received. The problem turned out to be that another label belonging to a nearby line was close to the line in question and *DesignAid* was mistakenly identifying it as being the questioned line's label. When the other label was moved a few spaces from the line in question, the problem disappeared. The error message would have been clearer if it had stated, for example, that *DesignAid* was trying to read two labels for one line.

5. **File Management**

There is no prompt to ask *DesignAid*'s user if he/she wants to save a file before deleting it from the screen. Much valuable work could be (and was!) lost as a result of a user forgetting to save a file after working on it.

When a file appears on the screen, it is bounded by file borders. These borders are easy to delete accidentally, and if they are deleted they cannot be brought back (easily), which means the file cannot be manipulated on the screen or deleted from it in the normal fashion. In order to do anything at this point, the author found it necessary to log completely out of *DesignAid*, then log back in. The screen will not contain the file being worked on at the time of logging out--the file must be called back onto the screen. Any changes made before the file border is erased will NOT be in the file.

No warnings exist to prevent a user-created program file from being overwritten by a report file. The author had generated a data dictionary report about the level one data flow diagram of this thesis and inadvertently saved it to the file name of the level one data flow diagram, which, of course, overwrote the data flow diagram the file contained. Reconstruction of the entire data flow diagram was necessary, as it had not been backed up at that point.

Most reports generated by the reporting process are saved to disk whether or not *DesignAid*'s user desires them to be saved. They clutter disk space and must be erased occasionally to keep the directory and disk uncluttered. A capability should exist to enable the user to decide which files to save to disk.

6. **Word Processing**

The word processing capabilities of *DesignAid* are rudimentary, but workable. It is not clear in the program documentation whether text files may be created with a more sophisticated word processing program and interfaced to *DesignAid.*

The paginate function particularly caused problems during the definition of a structure file to the dictionary, as the asterisk that was a component of the hard page break was not accepted by the definition process. The page break was located at the end of a file of text and, since it was preceded by an asterisk, the dictionary interpreted it as being a comment. An error message was produced saying that a comment cannot terminate a file. Validating data flow diagrams which had page breaks was not a problem, however.

B. EXCELERATOR

Excelerator's documentation was more user-friendly than was *DesignAid*'s in terms of explaining not only how to enter information into the data dictionary, but how that information is accessed and processed by the program. The writers of *Excelerator*'s documentation seemed to want to convey the program's basic functioning to its user, not just what keystrokes the user could make to enter and extract information about the system being designed. There are, however, a few peccadillos in *Excelerator* also.

1. **Tutorial/User's Guide/Reference Guide**

The Tutorial is generally well-written in a personable style. Enough general information is provided so that a good overall grasp of what Excelerator can do is obtained. One item mentioned in the tutorial was unclear, however: the tutorial writes that crossed connectors are to be avoided, but does not state that the program will not

function if they do cross each other. Discussion with a technical representative of InTech revealed that crossed connectors do enable *Excelerator* to function correctly, and are only discouraged for aesthetic reasons [Ref. 15].

The User's and Reference Guides are written in an understandable and concise manner, but do not explain what certain minor items are. For example, the explanation of the fields "Prompt", "Column Header", and "Short Header" in the Element Description Screen is that they are for documentation purposes only, with little, if any, explanation of how they can be used in documentation.

2. **Data Dictionary**

Entering information into the data dictionary was not a complicated process. It involved simply filling out a Description screen form of information, consisting minimally of the entity name. The data dictionary automatically keeps track of the location of the entity. Should the user desire to enter more information about entities such as data stores or data flows, a short series of screens can be easily filled in from inside the Description screen to more fully describe the entity. One notation tool that Yourdon teaches to record repetitions of the same data, the iteration notation, is not clearly supported by *Excelerator*'s description process (the occurrence field in a record screen is the closest comparison). *Excelerator* does not support Yourdon structured notation; the contents of data stores and data flows are in an indented format. Additionally, the description of entities to the dictionary must be done one at a time, whereas *DesignAid* can automatically describe an entire data flow diagram's objects (entities) at once.

An attractive feature of *Excelerator* is that the user can readily assess which names of a particular entity type (e.g., data flow) have been defined to the dictionary as he/she is getting ready to define a new entity of that type to the dictionary. A list of what processes, data flows, data stores, etc., have been defined to the data dictionary is available at a glance on the screen with the press of a key. This reminds *Excelerator*'s user what names have been used before so that the exact label desired can be created. *DesignAid* provides the same type of information, but it is more cumbersome to retrieve.

Pseudocode mini-specifications for lowest-level processes were difficult to prepare when the mini-specs were longer than the space in the Description screen allowed. Discussion with the technical representative suggested that *Excelerator* supports mini-specifications via structure charts rather than pseudocode [Ref. 15].

3. **Diagramming**

Diagramming in *Excelerator* at first seemed easier than diagramming in *DesignAid* because a mouse was used. However, small annoyances combined to make getting used to the way *Excelerator* draws diagrams a little more difficult than to the way *DesignAid* does.

The three zoom levels *Excelerator* supports work only for the objects in the diagram, not for their labels. As a consequence, in layout mode (when the objects are smallest and all are visible onscreen), labels overlay objects and other labels around them, making it difficult to read and interpret the diagram. To see all labels clearly and in the exact position they will print in hardcopy, it is necessary to zoom to closeup mode, which loses sight of the diagram as a whole. *DesignAid* deals with this situation better in that in the two zoom modes it supports, objects and labels are kept in the same proportions, enabling the user to clearly comprehend the diagram at a glance.

The diagram, to fit on one page of paper when printed, must be drawn within the borders of one screen in medium zoom mode. The author did not find this obvious in the documentation, and discovered this phenomenon through trial and error on a large data flow diagram which had to be redrawn several times to obtain the correct output proportions.

Excelerator's user has the ability to define either what shape the labels of the objects in the diagram should have or to accept the shape *Excelerator* provides. The ability of the user to define the label shape is limited and depends on where in the diagram the label is to be placed. How the label appears on the screen also is determined by how it is defined to the Description screen in the data dictionary, and how the label is defined to the Description screen takes precedence over the user's label definiton without warning the user.

Excelerator allows three sizes of text to be used in text blocks on a diagram: small, medium, and large. All three sizes show up on the screen, but only the small and large sizes print on an *Epson FX* printer, with no readily-discernable explanation in the documentation about why the medium size text does not print.

Excelerator does not allow areas of a data flow diagram to be moved; objects and their connectors must be moved one at a time.

4. **Validation**

Balancing problems surfaced when the input to a parent data flow diagram process did not match the inputs to the child data flow diagram (the inputs to the child

data flow diagram were elements of the input to the parent process). It was not immediately obvious to the author that the interface command in the data flow diagram graphics menu was to be used to indicate data flowing to another data flow diagram. The author initially used offpage connectors to indicate data flowing to another diagram, but, after a conversation with an InTech representative, and upon close inspection of the documentation, the error became obvious. The Reference Guide is unclear about what an off-page connector is used for in *Excelerator*'s Yourdon notation.

5. **File Management**

File Management in *Excelerator* is generally good. Prompts remind the user to save recent work before exiting from the file, an invaluable aid for forgetful users. To inspect another file, however, it is necessary to get out of the file currently open and open the file to be inspected, whereas *DesignAid* allows the file to be inspected to be opened while inside the current open file, which is a time saver. However, as mentioned in the *DesignAid* analysis subsection of this thesis, *DesignAid*'s file borders are sometimes lost, making this capability one which could lose its attractiveness to a user who is not careful to keep his/her cursor between file borders.

Printing is slow compared to *DesignAid*'s comparable quality print.

6. **Word Processing**

Excelerator, like *DesignAid*, supports rudimentary word processing capabilities. However, unlike *DesignAid*, *Excelerator* can link to other word processing programs to create text files which can be stored with the rest of the project data in the project directory. When it is time to print the documentation for the project, the text files can be printed out with other *Excelerator* files.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Both products, once an analyst and a system's end user are fully knowledgeable of and comfortable with how they work, appear to be good vehicles for expressing the logical model of how a system works or should work. Both products definitely decrease the time it takes to draw data flow diagrams from manual methods. Once the data flow diagrams are drawn, the computerized validation process decreases the time spent searching for syntactical and balancing errors in the data flow diagrams. A thorough understanding of how the product searches for errors and what the error messages received mean is necessary to make full use of the product. Not surprisingly, understanding each product takes education, time, and experience, but once the idiosyncrasies of the products are overcome by familiarity with them, they are both relatively easy to use.

The judicious use of a printer with each product to print out changes to data flow diagrams is extremely helpful--even necessary--to obtain the quickest access to data flow diagrams other than the one currently onscreen. Switching from one data flow diagram to another onscreen diverts the product user's attention at least momentarily, and often trains of thought are annoyingly interrupted (and sometimes derailed). One interesting solution to this problem, suggested by a person who did systems analysis and design manually in the past, is to post printouts of all data flow diagrams and data dictionary contents on a large wall in a central location. Comparison of data flow diagrams can be made at a glance with this method, and thought processes are not interrupted.

The total size of an Abbreviated Systems Decision Paper is increased using the output of the two CASE products discussed in this thesis. The clarity, however, is much improved for the ASDP reader who understands how to read the output, and a picture does seem to be worth a thousand words. A series of these "pictures", as illustrated in Appendices C and D, could accurately and feasibly replace the current DoD requirement for a textual description of at least the current logical model of a system. As can be observed, the tradeoff seems to be increased size for increased understanding.

To keep the size of an ASDP from becoming onerous, and to increase understanding of the present system, a presentation diagram of the current physical model could replace or augment the current logical model in the ASDP. This would associate physical images with logical processes in the mind of the reader, and perhaps increase comprehensibility of the system being designed. This idea is worthy of further exploration.

B. RECOMMENDATIONS

Several recommendations can be made based on the results of this thesis:

- that CASE tools continue to be used by people who are experienced in systems analysis and design and familiar with CASE tools to document at least logical models of systems being designed for DoD,
- that a continuing education program be in place to educate not only personnel who will actually use CASE products to create specifications for DoD systems, but also those personnel who will read and act on the output from these products,
- that close liaison with the company/corporation producing the CASE product be maintained, and as much formal education as possible be obtained to keep personnel abreast of the expanding capabilities of CASE products, and
- that output from CASE products, appropriate to the individual system being designed, replace Section 4.1 of the Abbreviated Systems Decision Paper.

APPENDIX A
DESCRIPTION OF TERMS USED IN SCHEDULING PROCESS

Final Exam Period (or Exam Period). During final exam week, final exams are scheduled for two-hour time periods, three times daily (0800-1000, 1000-1200, and 1400-1600), from Monday through Thursday. A total of 12 final exam periods (3/day * 4 days) are available in which to schedule final exams.

Refresher Courses. Refresher courses are taught during the last six weeks of certain quarters during the year. The rooms in which they are taught are not available for scheduling final exams, and the professors teaching them are not available to give final exams during the times they are taught.

Section. Student who are scheduled for the same sequence of courses in a particular quarter comprise a section. A section is the smallest unit to be scheduled for a course. Sections vary in size depending upon how many students happen to request the same set of courses, and by chance be scheduled for the same course segments in the same sequence.

Segment. If more than the optimal number of students one classroom contains request a course, the course may be split into segments to accommodate the overflow as appropriate. These may be taught by one or more faculty members. The course content remains the same in all segments, and the course number is augmented with a segment number to distinguish it from other segments.

Room. A total of 110 rooms with capacities from 10 to 80 people are available in six buildings on campus. Courses offered by a particular department are usually taught within a specified building, as per 3, and final exams are usually scheduled in the same building as the course is taught.

APPENDIX B
FORMAT OF AN ABBREVIATED SYSTEM DECISION PAPER

SECTION 1 *MISSION NEED*

1.1 *NEED.* Outline the need for automation as related to specific elements of the organization's mission. Clearly identify problems and describe their relationship to the mission of the organization for which the system will be developed.

1.2 *PRIORITY.* Describe the relative priority of the need to other mission needs of the organization.

SECTION 2 *REQUIRED CAPABILITIES*

2.1 *USER REQUIREMENTS.* Describe user requirements in functional terms.

2.2 *PERFORMANCE REQUIREMENTS.* Identify the standards by which the performance of the IS is to be measured and the minimum standard of acceptable performance. These standards should be quantifiable and demonstrably measurable.

2.3 *INTERFACE REQUIREMENTS.* Describe the proposed ISs relationship with existing or propoesd systems. Include the purpose of the requirement for the interface and manner the interface is to be achieved.

2.4 *COMMUNICATIONS REQUIREMENTS.* Describe all potential communication support requirements to include projected volumes and types of data to be exchanged and the frequency of data exchange.

2.5 *CLASSIFICATION REQUIREMENTS.* Describe the requirements for classified processing.

2.6 *OPERATING ENVIRONMENT.* Identify the operating environment in which the IS must operate. Address the requirements for the IS to operate in a deployed environment.

SECTION 3 *PROPOSED ALTERNATIVE*

3.1 *GENERAL.* Provide a summary of the preferred alternative to meet the need. Identify any assumptions or constraints considered in the selection.

SECTION 4 *OTHER ALTERNATIVES*

4.1 *CURRENT SYSTEM.* Summarize the current system. (note: see Appendices C and D for current system described in this thesis.)

4.2 *OTHER ALTERNATIVES CONSIDERED.* Summarize all other alternatives considered and explain why each was not selected as a proposed solution. This discussion should center on the technical and operational aspects of each alternative.

SECTION 5 *COST ANALYSIS*

5.1 *STATEMENT OF COSTS*

a. Total costs for each year will be identified by appropriation (i.e., RDT&E, PMC, O&MMC, MCON, etc.) for each alternative using the following guidelines:

	ONE-TIME COSTS		*RECURRING COSTS*	
	ADP	*NON-ADP*	*ADP*	*NON-ADP*
Personnel:				
Organizational				
Contractor				
Training				
TAD				
Equipment:				
Procurement				
Lease				
Maintenance				
Installation				
Site Preparation				
Telecommunications				
Software:				
Development				
Procurement				
Lease				
Maintenance				
Telecommunications				
Other				
TOTAL				

b. Costs will be summarized for each alternative in the following manner:

	ONE-TIME COSTS		*RECURRING COSTS*	
	ADP	*NON-ADP*	*ADP*	*NON-ADP*
Period				
1				
2				
3				
4				
N				
TOTAL				

SECTION 6 *BENEFIT ANALYSIS*

6.1 *GENERAL.* Benefits, for this purpose, are beneficial effects on the mission effectiveness of the proposed IS. All benefits that can be identified should be listed and discussed for the proposed alternative.

SECTION 7 *FUNDING*

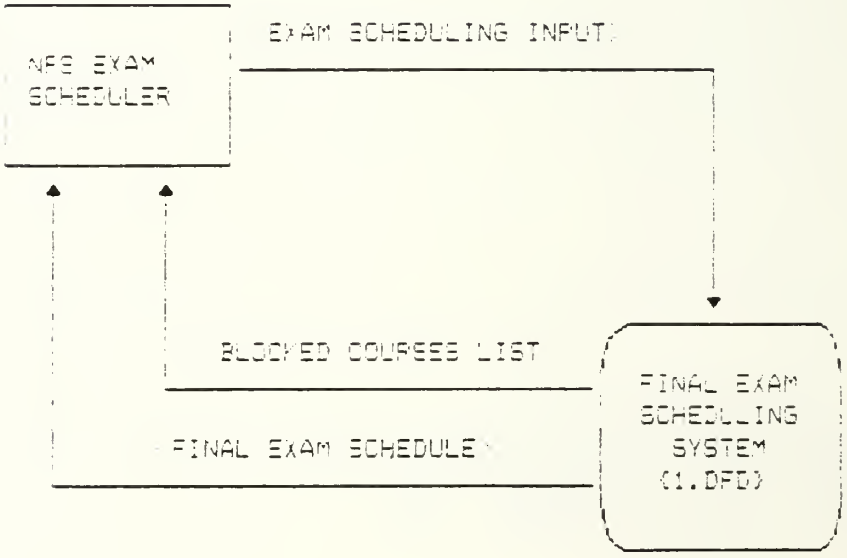
7.1 *GENERAL.* A statement regarding the availability of funding to support the life cycle costs of the proposed IS should be included. Identify the source and type of funding.

SECTION 8 *PLANNING DATA*

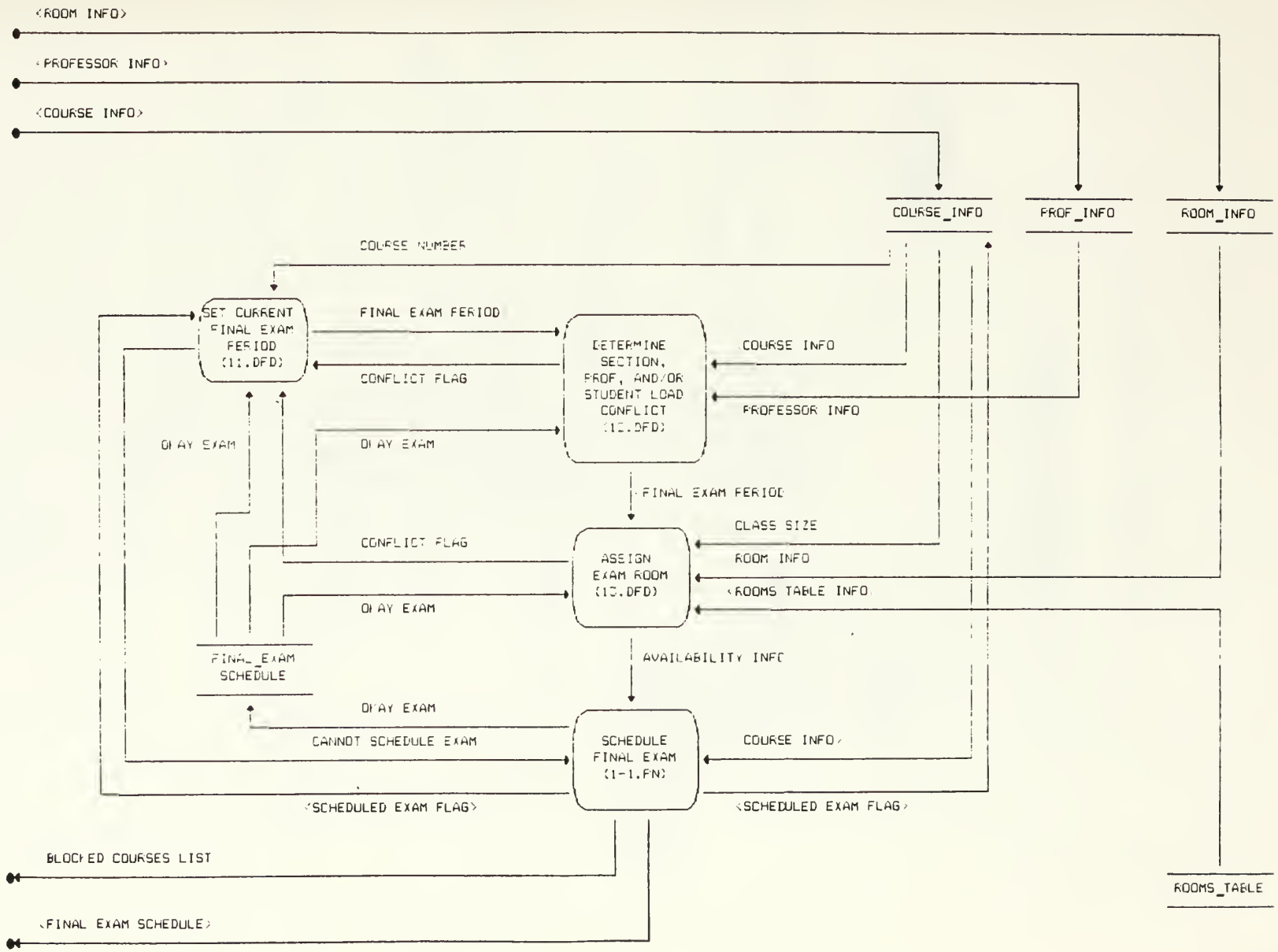
8.1 *GENERAL.* A discussion, if any, of the equipment considered in the analysis should be included. Indicate a milestone schedule to include dates for contract award, delivery of equipment and implementation of the IS.

APPENDIX C
SECTION 4.1 OF ASDP USING DESIGNAID

CONTEXT (0 LEVEL) DATA FLOW DIAGRAM
FOR
CURRENT LOGICAL MODEL OF NPS FINAL EXAM SCHEDULING SYSTEM

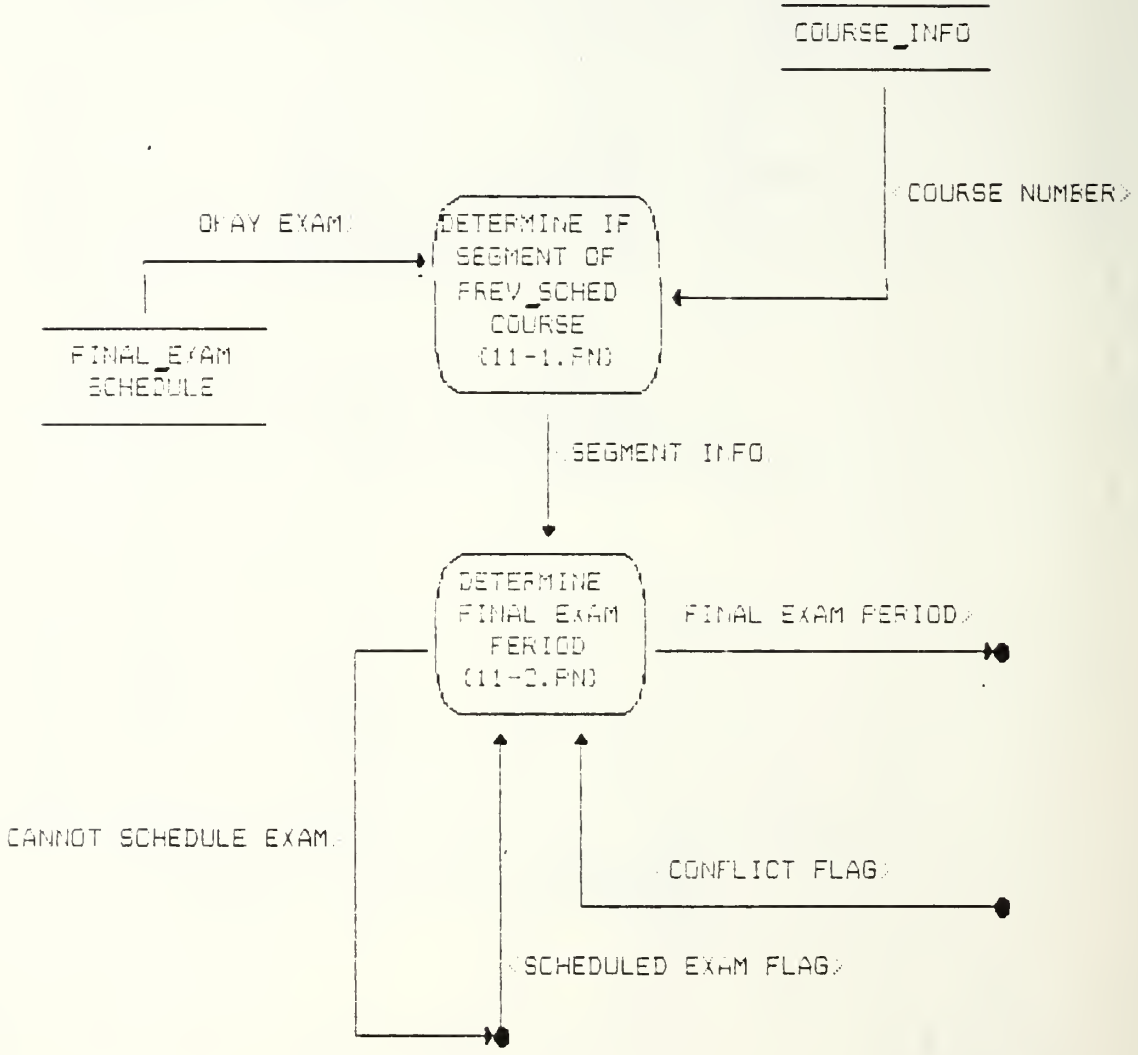


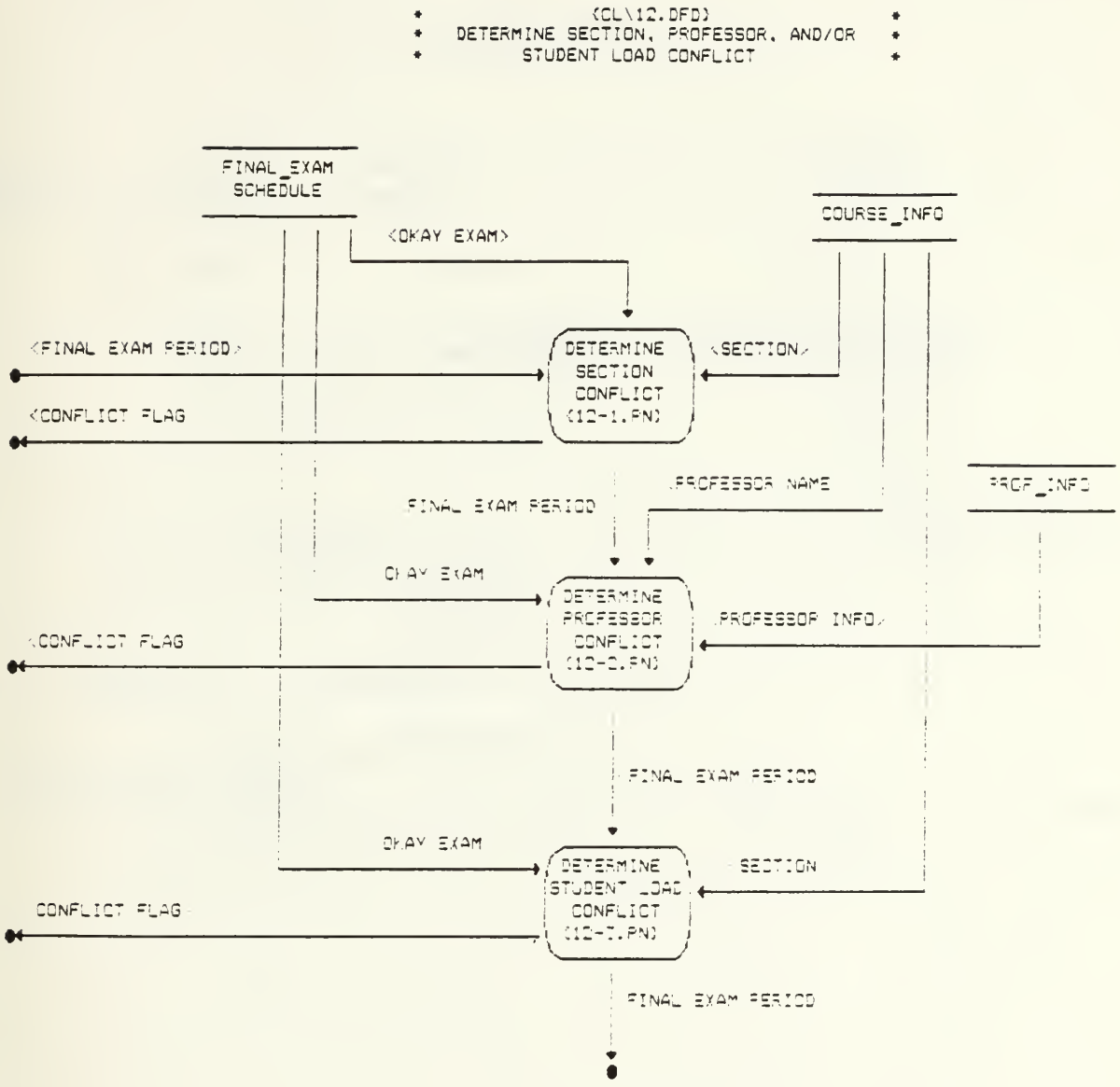
\CL\1.DFD\
LEVEL 1 DATA FLOW DIAGRAM
FOR
CURRENT LOGICAL MODEL OF NPS FINAL EXAM SCHEDULING SYSTEM

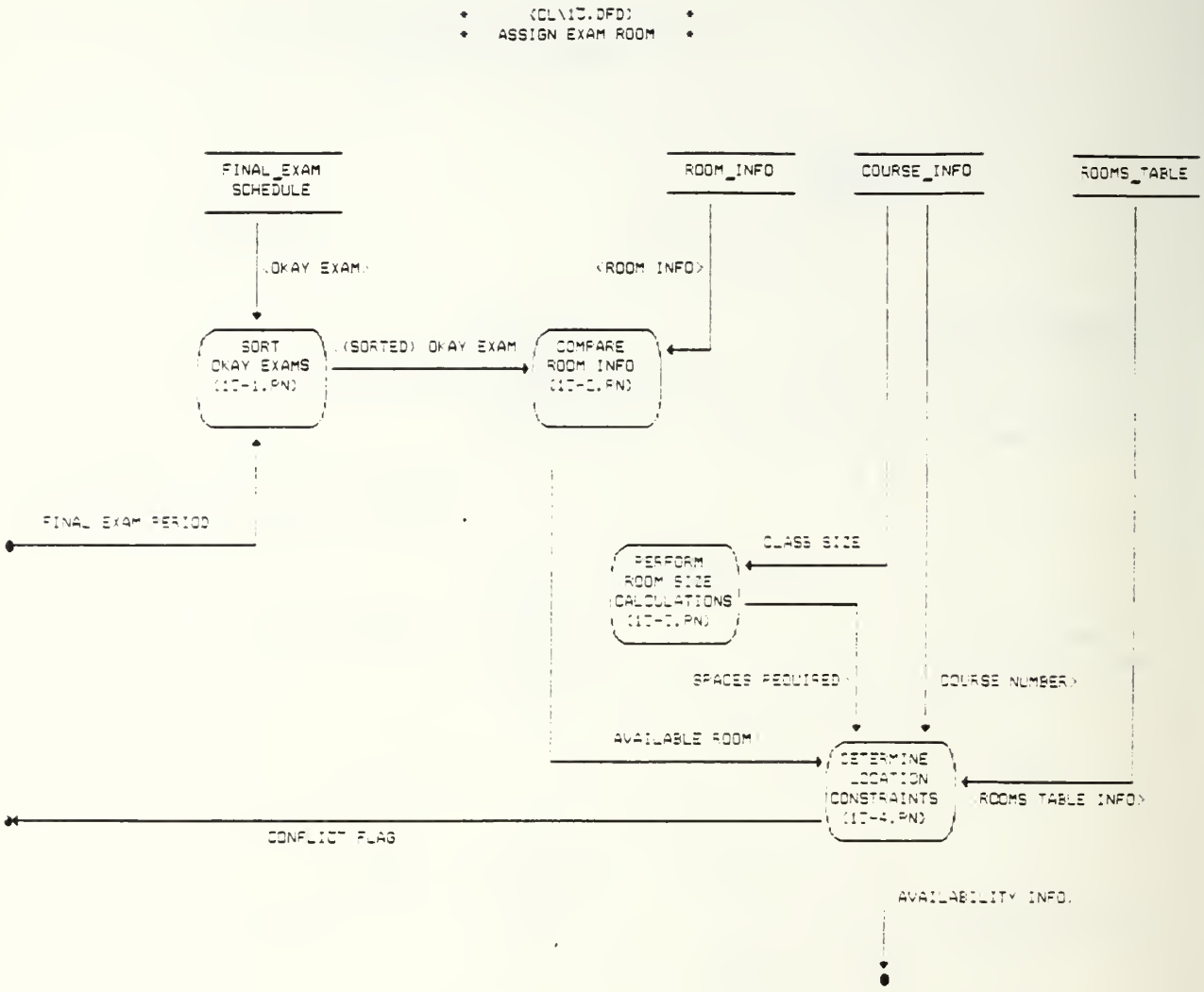


(CL\11.DFD)

SET CURRENT FINAL EXAM PERIOD




(CL\12.DFD)
DETERMINE SECTION, PROFESSOR, AND/OR
STUDENT LOAD CONFLICT
FINAL_EXAM
SCHEDULE
COURSE_INFO
<OKAY EXAM>
<FINAL EXAM PERIOD>
DETERMINE
SECTION
CONFLICT
(12-1.PN)
<SECTION>
<CONFLICT FLAG
<PROFESSOR NAME
PROF_INFO
FINAL EXAM PERIOD
OKAY EXAM
DETERMINE
PROFESSOR
CONFLICT
(12-2.PN)
<PROFESSOR INFO>
<CONFLICT FLAG
FINAL EXAM PERIOD
OKAY EXAM
DETERMINE
STUDENT LOAD
CONFLICT
(12-3.PN)
SECTION
CONFLICT FLAG
FINAL EXAM PERIOD


(CL\IC.DFD)
ASSIGN EXAM ROOM
FINAL_EXAM SCHEDULE
ROOM_INFO
COURSE_INFO
ROOMS_TABLE
(OKAY EXAM)
(ROOM INFO)
SORT OKAY EXAMS (IC-1.PN)
(SORTED) OKAY EXAM
COMPARE ROOM INFO (IC-2.PN)
FINAL EXAM PERIOD
CLASS SIZE
PERFORM ROOM SIZE CALCULATIONS (IC-3.PN)
SPACES REQUIRED
COURSE NUMBER
AVAILABLE ROOM
DETERMINE LOCATION CONSTRAINTS (IC-4.PN)
(ROOMS TABLE INFO)
CONFLICT FLAG
AVAILABILITY INFO.

```
*        (CL\1-1.PN)        *
*   SCHEDULE FINAL EXAM     *
```

Read AVAILABILITY INFO.

Read COURSE INFO.

Combine the appropriate elements of AVAILABILITY INFO and COURSE INFO to make OKAY EXAM and send to FINAL_EXAM SCHEDULE data store.
Send SCHEDULED EXAM FLAG = YES for current course to COURSE_INFO data store

If CANNOT SCHEDULE EXAM flag is received from SET CURRENT FINAL EXAM process
Then send SCHEDULED EXAM FLAG = NO for current course to COURSE_INFO data store.

At end-of-file of COURSE_INFO data store
Read contents of COURSE_INFO data store.

If SCHEDULED EXAM FLAG = NO
Then COURSE INFO is sent to BLOCKED COURSES LIST.

If SCHEDULED EXAM FLAG = YES
Then COURSE INFO is sent to FINAL EXAM SCHEDULE.

```
*              (CL\11-1.PN)              *
*       DETERMINE IF SEGMENT OF          *
*    PREVIOUSLY-SCHEDULED COURSE         *
```

Read COURSE NUMBER for the current course exam to be scheduled.

Read COURSE NUMBERs for all OKAY EXAMs.

If COURSE NUMBER of current course is a segment of COURSE NUMBER of OKAY EXAM
Then SEGMENT INFO = FINAL EXAM PERIOD of OKAY EXAM.

If COURSE NUMBER of current course is not a segment of COURSE NUMBER of OKAY EXAM
Then SEGMENT INFO = NO.

```
*            {CL\11-2.PN}            *
*   DETERMINE FINAL EXAM PERIOD      *


Set FINAL EXAM PERIOD = 0.

Repeat
  Read SEGMENT INFO.

    If SEGMENT INFO = FINAL EXAM PERIOD of OKAY EXAM
    Then FINAL EXAM PERIOD = FINAL EXAM PERIOD of OKAY EXAM.

    If SEGMENT INFO = NO
    Then FINAL EXAM PERIOD = FINAL EXAM PERIOD + 1.

  Repeat
    Read CONFLICT FLAG.

      Set Increment Counter = 0.

      If CONFLICT FLAG = CONFLICT
      and SEGMENT INFO = FINAL EXAM PERIOD of OKAY EXAM
      Then CANNOT SCHEDULE EXAM = NO GO.

      If CONFLICT FLAG = CONFLICT
      and SEGMENT INFO NO
      Then FINAL EXAM PERIOD = FINAL EXAM PERIOD + 1.

      Increment Counter = Increment Counter + 1.

  Until
    SCHEDULED EXAM FLAG = OK
    or Increment Counter = 10.
       *(That is, either the course's exam was successfully *
       * scheduled or all of the possible 10 final exam     *
       * periods have been tried unsuccessfully.)           *

  If INCREMENT COUNTER = 10
  Then CANNOT SCHEDULE EXAM = NO GO.


Until
  End of COURSE_INFO data store.
```

```
*          (CL\12-1.PN)          *
*  DETERMINE SECTION CONFLICT    *


Read FINAL EXAM PERIOD.

Read OKAY EXAMs.

Read SECTION(s).

   If FINAL EXAM PERIOD = FINAL EXAM PERIOD of OKAY EXAM
   Then compare current course s SECTIONs to OKAY EXAM's
     SECTIONs.

      If at least one of the current course s sections
         match with any of OKAY EXAM s SECTIONs
      Then set CONFLICT FLAG = CONFLICT and pass control to
         process SET CURRENT FINAL EXAM PERIOD.

      If none of the current course s sections match with
         OKAY EXAM s SECTIONs
      Then pass FINAL EXAM PERIOD to next process
         (DETERMINE PROFESSOR CONFLICT).
```

```
*              (CL\12-2.PN)              *
*   DETERMINE PROFESSOR CONFLICT         *
```

Read FINAL EXAM PERIOD.

Read PROFESSOR NAME.

Read OKAY EXAM.

Gather all OKAY EXAMS whose FINAL EXAM PERIOD equals FINAL EXAM PERIOD read from previous process.

On these, check for PROFESSOR NAME being the same as PROFESSOR NAME for current course whose exam is being scheduled.

If a match is made
Then set CONFLICT FLAG = CONFLICT and pass control to process SET CURRENT FINAL EXAM PERIOD.

If a match is not made
Then read PROFESSOR INFO.

If COURSE TIME of TEACHING SCHEDULE is the same as (or is included in) the current FINAL EXAM PERIOD
and REFRESHER FLAG = YES
Then set CONFLICT FLAG = CONFLICT and pass control to process SET CURRENT FINAL EXAM PERIOD.

Else pass current FINAL EXAM PERIOD to next process (DETERMINE STUDENT LOAD CONFLICT)

If there are no OKAY EXAMs whose FINAL EXAM PERIOD = current FINAL EXAM PERIOD
Then pass current FINAL EXAM PERIOD to next process (DETERMINE STUDENT LOAD CONFLICT).

```
*                (CL\12-3.PN)                *
*   DETERMINE STUDENT LOAD CONFLICT          *
```

Read FINAL EXAM PERIOD.

Read OKAY EXAM.

Read SECTION of current course.

Gather all okay exams which are scheduled during plus or minus 1 FINAL EXAM PERIOD of the current FINAL EXAM PERIOD.

On these, compare SECTIONs of OKAY EXAMs to SECTIONs of current course.

If there are any matches
Then set CONFLICT FLAG = CONFLICT and pass control to process SET CURRENT FINAL EXAM PERIOD.

If there are no matches of SECTIONs
Then gather all OKAY EXAMs scheduled during one day (periods 1 through 3, 4 through 6, 7 through 9, and 10 through 12.)

Compare SECTIONs of current course to SECTIONs of OKAY EXAMs.

If any SECTION of current course appears twice during the day
Then set CONFLICT FLAG = CONFLICT and pass control to process SET CURRENT FINAL EXAM PERIOD.

Else pass FINAL EXAM PERIOD to next process (ASSIGN EXAM ROOM).

```
*     (CL\13-1.PN)      *
*   SORT OKAY EXAMS     *
```

Read all OKAY EXAMs.

Send all OKAY EXAMs whose FINAL EXAM PERIOD equals the current FINAL EXAM PERIOD to the next process (COMPARE ROOM INFO).

```
*      (CL\13-2.PN)     *
*  COMPARE ROOM INFO    *
```

Read ROOM NUMBER of (SORTED) OKAY EXAM.

Read ROOM INFO.

Perform a match of ROOM NUMBER of (SORTED) OKAY EXAM and ROOM INFO for each room in the ROOM_INFO data store.

Send unmatched rooms (AVAILABLE ROOM) to next process (DETERMINE LOCATION CONSTRAINTS).

* (CL\13-3.PN) *
* PERFORM ROOM SIZE CALCULATIONS *

Read CLASS SIZE.

CLASS SIZE * 1.5 = SPACES REQUIRED

Pass SPACES REQUIRED to next process (DETERMINE LOCATION CONSTRAINTS).

* (CL\13-4.PN) *
* DETERMINE LOCATION CONSTRAINTS *

Read COURSE NUMBER.

Read SPACES REQUIRED.

Read AVAILABLE ROOMs.

Read ROOMS TABLE INFO.

Based on COURSE NUMBER and AVAILABLE ROOMs, determine which room(s) in ROOMS TABLE INFO will satisfy room constraints (such as SPACES REQUIRED, preferred locations, refresher classes, same-floor requirement in case more than one room is needed, etc.).

Pass AVAILABILITY INFO to next process (SCHEDULE FINAL EXAM).

OR

If no rooms are available after all constraints have been considered, set CONFLICT FLAG = CONFLICT and pass control to process SET CURRENT FINAL EXAM PERIOD.

```
AVAILABILITY INFO =
      { AVAILABLE ROOM }
    + FINAL EXAM PERIOD

BLOCKED COURSES LIST =
      { COURSE INFO }

COURSE INFO =
        COURSE NUMBER
    + PROFESSOR NAME
    +{ SECTION }
    + CLASS SIZE

EXAM SCHEDULING INPUT =
      { COURSE INFO }
    +{ PROFESSOR INFO }
    +{ ROOM INFO }

FINAL EXAM SCHEDULE =
      { OKAY EXAM }

OKAY EXAM =
        COURSE NUMBER
    + PROFESSOR NAME
    +{ SECTION }
    + FINAL EXAM PERIOD
    + ROOM NUMBER

PROFESSOR INFO =
        PROFESSOR NAME
    +{ TEACHING SCHEDULE }

ROOM INFO =
        ROOM NUMBER
    +{ REFRESHER TIME }
    + ROOM SIZE

ROOMS TABLE INFO =
        COURSE INDICATOR
    + BUILDING
    + FLOOR
    + ROOM SIZE

SORTED OKAY EXAM =
      { OKAY EXAM }

TEACHING SCHEDULE =
        COURSE NUMBER
    + COURSE TIME
    + REFRESHER FLAG
```



```
COURSE_INFO =
      { COURSE INFO }
    + SCHEDULED EXAM FLAG

FINAL_EXAM SCHEDULE =
      { OKAY EXAM }

PROF_INFO =
      { PROFESSOR INFO }

ROOMS_TABLE =
      { ROOMS TABLE INFO }

ROOM_INFO =
      { ROOM INFO }
```

SEARCH CRITERIA
OBJECT TYPE DATA ELEMENT

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
CLASS SIZE	DATA ELEMENT	NUMERIC		I - 150

FILE NAME
C:\DONNA\CL\1.DFD
C:\DONNA\CL\13.DFD
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
FINAL EXAM PERIOD	DATA ELEMENT	NUMERIC		I - 12

FILE NAME
C:\DONNA\CL\1.STR
C:\DONNA\CL\1.DFD
C:\DONNA\CL\11.DFD
C:\DONNA\CL\13.DFD
C:\DONNA\CL\12.DFD
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
CONFLICT FLAG	DATA ELEMENT	ALPHABETIC		SECT, PROF, STUDENT LOAD

FILE NAME
C:\DONNA\CL\1.DFD
C:\DONNA\CL\11.DFD
C:\DONNA\CL\13.DFD
C:\DONNA\CL\12.DFD

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
CANNOT SCHEDULE EXAM	DATA ELEMENT	ALPHABETIC		NO GO

FILE NAME
C:\DONNA\CL\1.DFD
C:\DONNA\CL\11.DFD

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
SCHEDULED EXAM FLAG	DATA ELEMENT	ALPHABETIC		OK

FILE NAME
C:\DONNA\CL\1.STR
C:\DONNA\CL\1.DFD
C:\DONNA\CL\11.DFD

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
ROOM NUMBER	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\0.STR
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
AVAILABLE ROOM	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\1.STR
C:\DONNA\CL\13.DFD

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
PROFESSOR NAME	DATA ELEMENT	ALPHABETIC		

FILE NAME
C:\DONNA\CL\12.DFD
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
SECTION	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\12.DFD
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
REFRESHER TIME	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
AVAILABLE FLAG	DATA ELEMENT	ALPHABETIC		YES, NO

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
COURSE TIME	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
REFRESHER FLAG	DATA ELEMENT	ALPHABETIC		YES, NO

FILE NAME
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
SEGMENT INFO	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\11.DFD

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
ROOM SIZE	DATA ELEMENT	NUMERIC		

FILE NAME
C:\DONNA\CL\1.STR
C:\DONNA\CL\0.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
COURSE INDICATOR	DATA ELEMENT	ALPHABETIC		

FILE NAME
C:\DONNA\CL\1.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
BUILDING	DATA ELEMENT	ALPHANUMERIC		

FILE NAME
C:\DONNA\CL\1.STR

OBJECT NAME	OBJECT TYPE	DATA TYPE	INITIAL VALUE	VALUE CONSTRAINTS
FLOOR	DATA ELEMENT	NUMERIC		

FILE NAME

OBJECT NAME: AVAILABLE FLAG OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/08/87
DATA TYPE: ALPHABETIC VALUE CONST: YES, NO DATE MODIFIED:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: AVAILABLE ROOM OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/08/87
DATA TYPE: ALPHANUMERIC VALUE CONST: DATE MODIFIED:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: BUILDING OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/10/87
DATA TYPE: ALPHANUMERIC VALUE CONST: DATE MODIFIED: 08/10/87
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: CANNOT SCHEDULE EXAM OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/08/87
DATA TYPE: ALPHABETIC VALUE CONST: NO GO DATE MODIFIED:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: CLASS SIZE OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/08/87
DATA TYPE: NUMERIC VALUE CONST: 1 - 150 DATE MODIFIED:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: CONFLICT FLAG OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/08/87
DATA TYPE: ALPHABETIC VALUE CONST: SECT, PROF, STUDENT LOAD DATE MODIFIED:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: COURSE INDICATOR OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/10/87
DATA TYPE: ALPHABETIC VALUE CONST: DATE MODIFIED: 08/10/87
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: COURSE TIME OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
ALIAS OF: DEFINED BY: donna DATE DEFINED: 08/08/87
DATA TYPE: ALPHANUMERIC VALUE CONST: DATE MODIFIED:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT NAME: FINAL EXAM PERIOD
ALIAS OF:
DATA TYPE: NUMERIC VALUE CONST: 1 - 1C
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

OBJECT NAME: FLOOR
ALIAS OF:
DATA TYPE: NUMERIC VALUE CONST:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/10/87
DATE MODIFIED:

OBJECT NAME: NO CONFLICTS
ALIAS OF:
DATA TYPE: ALPHABETIC VALUE CONST: OK
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

OBJECT NAME: PROFESSOR NAME
ALIAS OF:
DATA TYPE: ALPHABETIC VALUE CONST:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

OBJECT NAME: REFRESHER FLAG
ALIAS OF:
DATA TYPE: ALPHABETIC VALUE CONST: YES, NO
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

OBJECT NAME: REFRESHER TIME
ALIAS OF:
DATA TYPE: ALPHANUMERIC VALUE CONST:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

OBJECT NAME: ROOM NUMBER
ALIAS OF:
DATA TYPE: ALPHANUMERIC VALUE CONST: ,
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

OBJECT NAME: ROOM SIZE
ALIAS OF:
DATA TYPE: NUMERIC VALUE CONST:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:
OBJECT TYPE: DATA ELEMENT STATUS UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/10/87
DATE MODIFIED:

OBJECT NAME: SCHEDULED EXAM FLAG
ALIAS OF:
DATA TYPE: ALPHABETIC VALUE CONST: OK
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

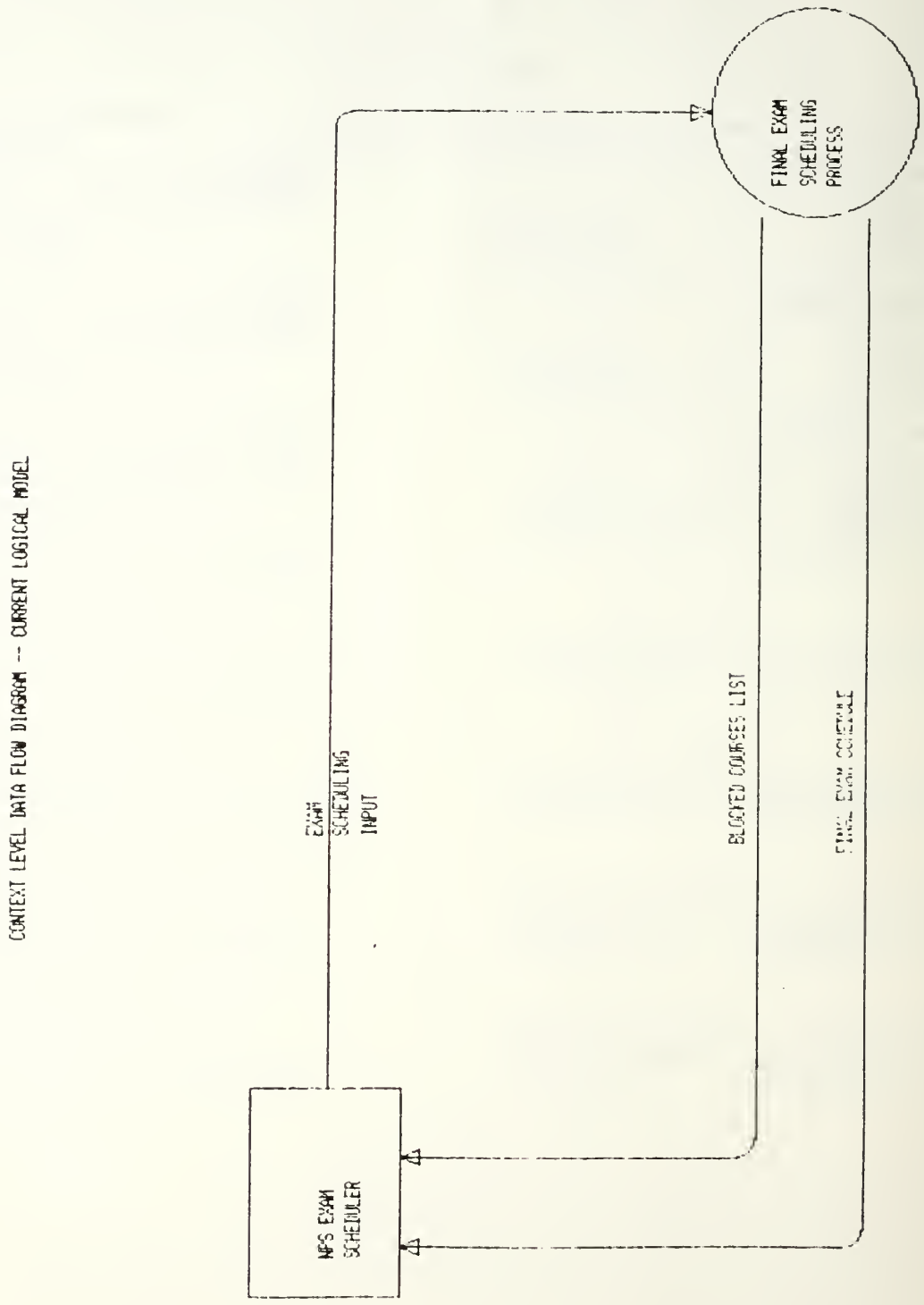
OBJECT NAME: SECTION
ALIAS OF:
DATA TYPE: ALPHANUMERIC VALUE CONST:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED: 08/08/87

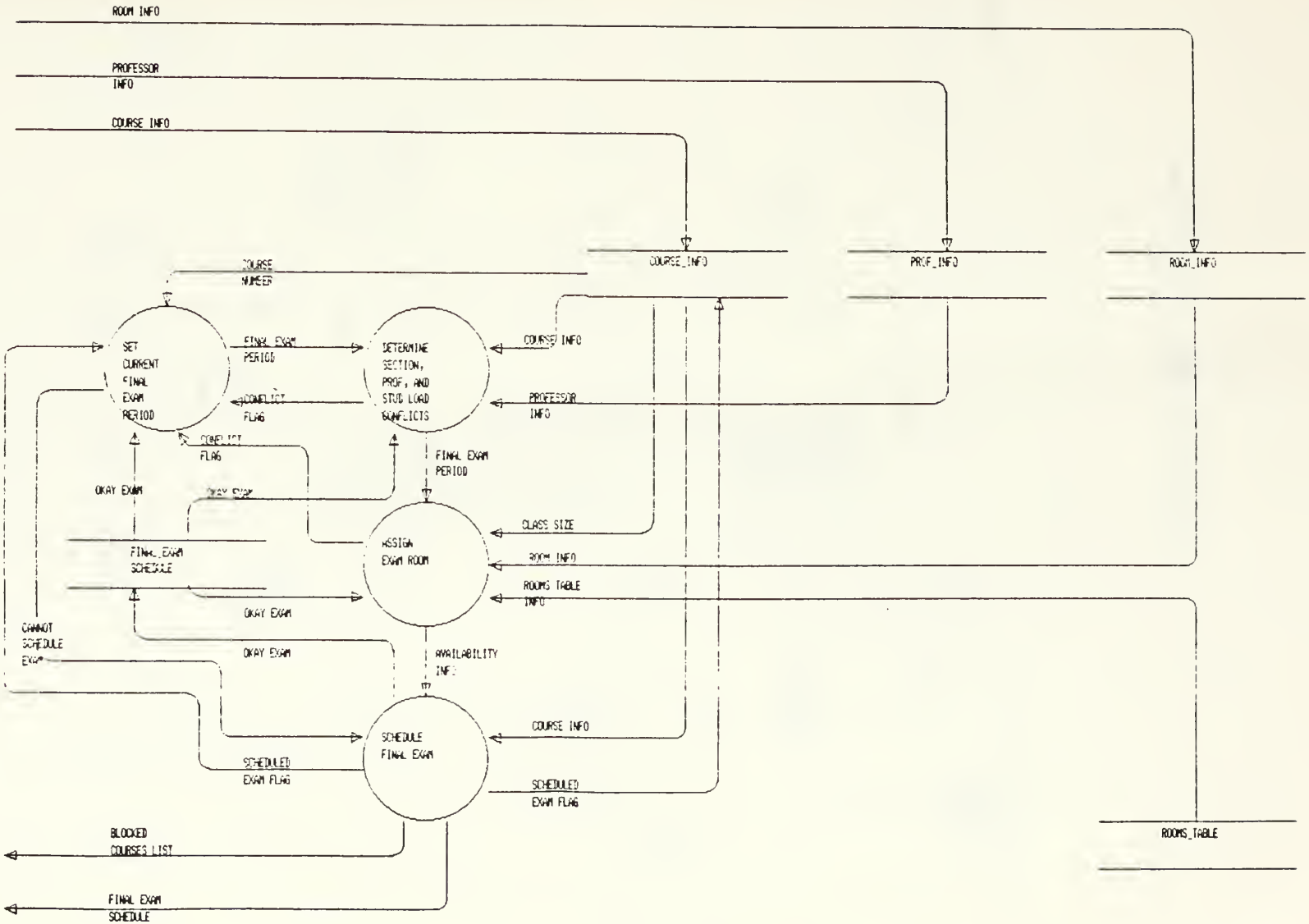
OBJECT NAME: SEGMENT INFO
ALIAS OF:
DATA TYPE: ALPHANUMERIC VALUE CONST:
DATA SIZE: INITIAL VALUE:
DESCRIPTION:

OBJECT TYPE: DATA ELEMENT STATUS: UNLOCKED
DEFINED BY: donna DATE DEFINED: 08/08/87
DATE MODIFIED:

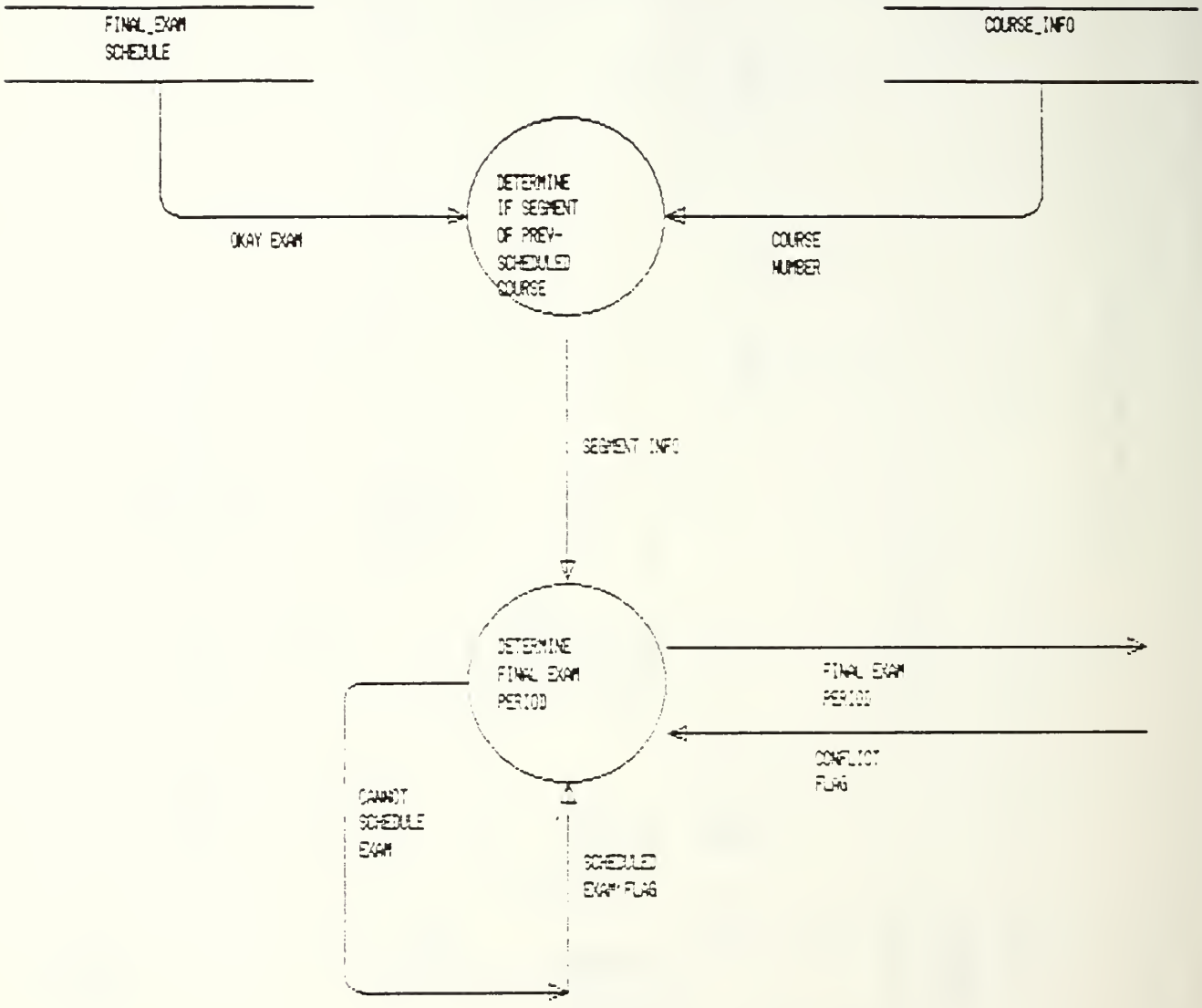
APPENDIX D
SECTION 4.1 OF ASDP USING EXCELERATOR



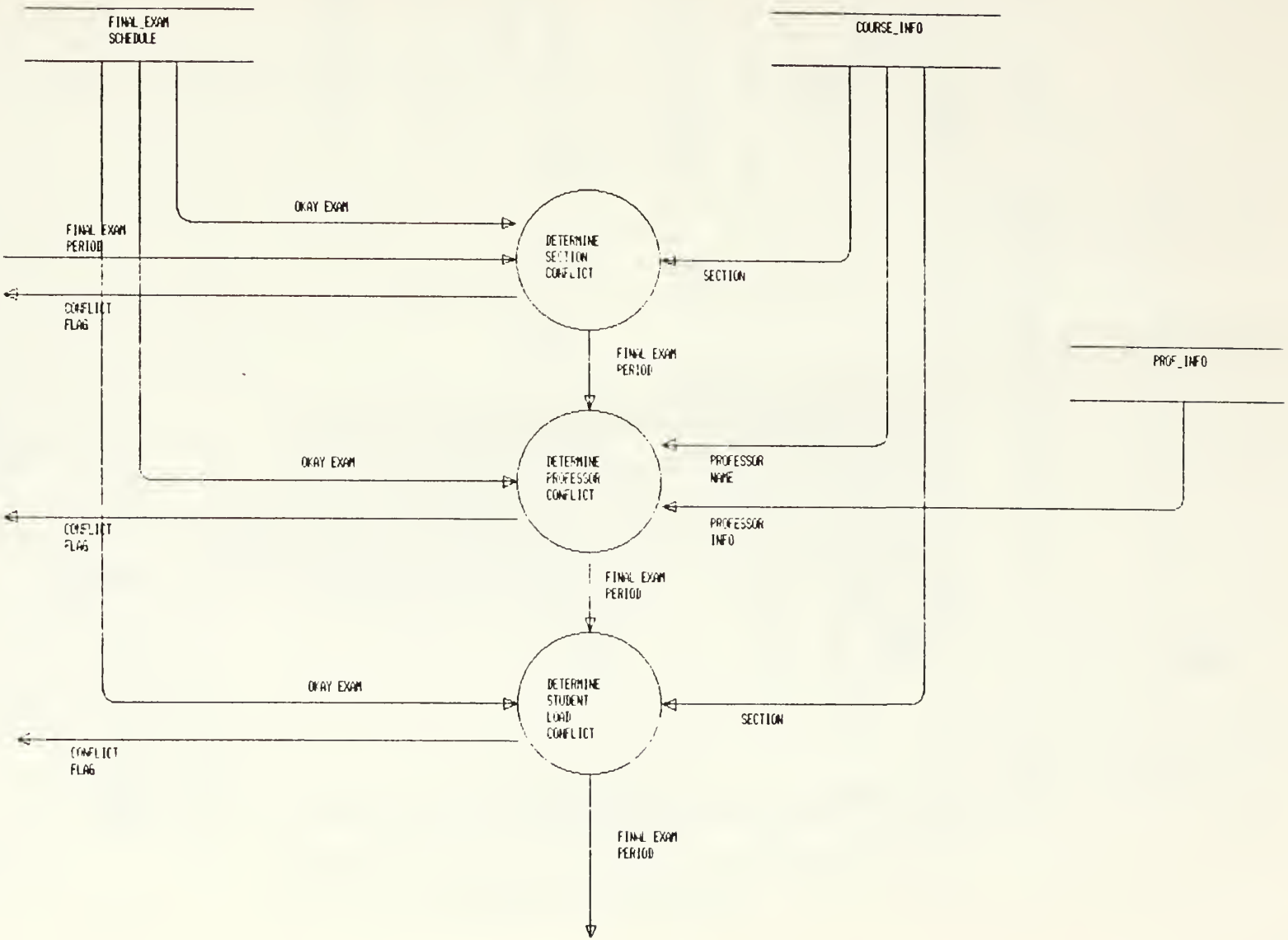
1 LEVEL DATA FLOW DIAGRAM — CURRENT LOGICAL MODEL



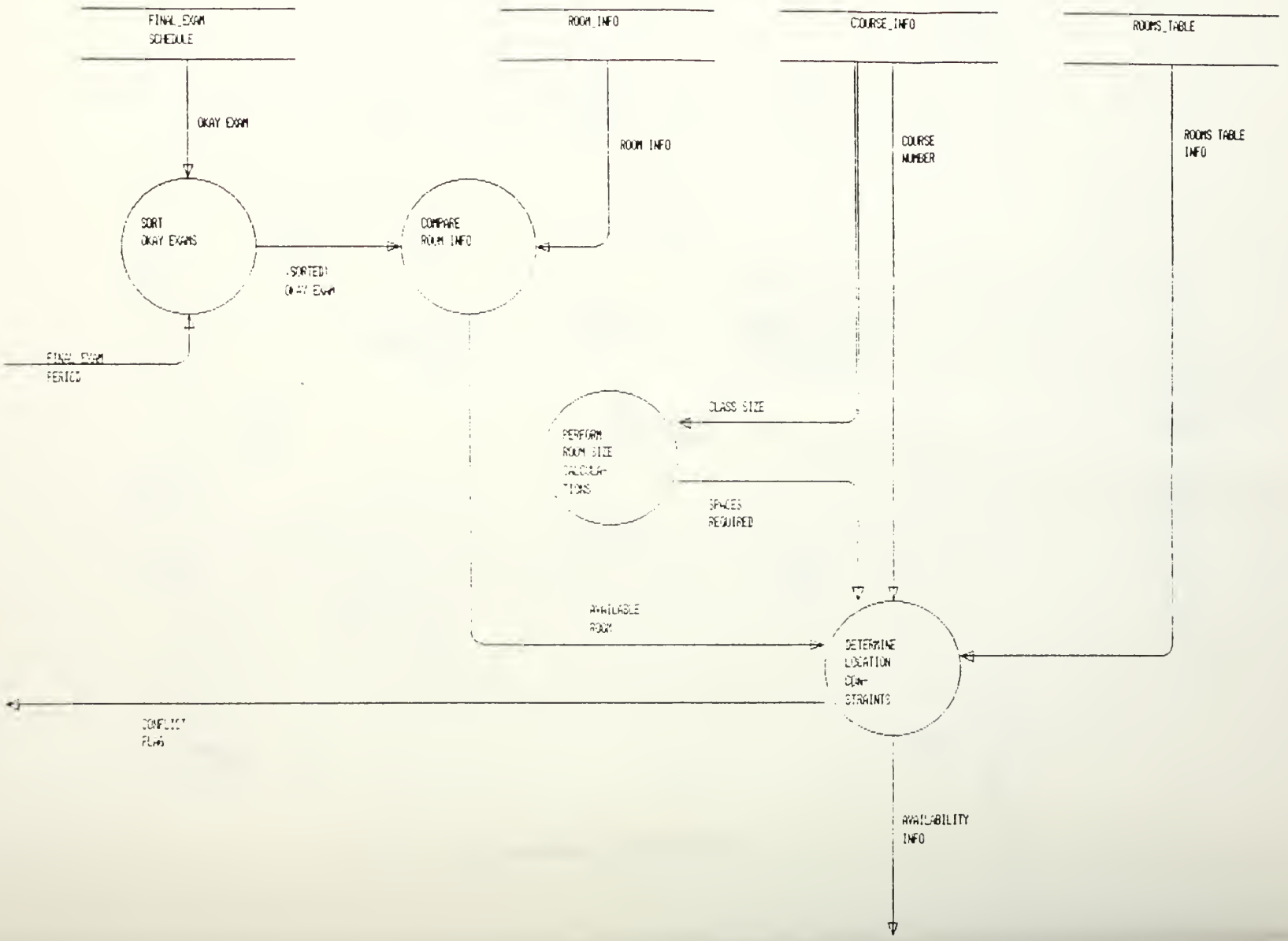
2.1 LEVEL DATA FLOW DIAGRAM — CURRENT LOGICAL MODEL



2.2 LEVEL DATA FLOW DIAGRAM -- CURRENT LOGICAL MODEL



2.3 LEVEL DATA FLOW DIAGRAM — CURRENT LOGICAL MODEL



DATE: 2-SEP-87 REC CONTAINS ANY - SUMMARY OUTPUT PAGE 1
TIME: 13:57 NAME: * EXCELERATOR 1.7

Record Contains Any Entity Type		Occ	Seq
AVAILABILITY INFO	ELE AVAILABLE ROOM	400	000
	ELE FINAL EXAM PERIOD	001	000
BLOCKED COURSES LIST	REC COURSE INFO	001	000
COURSE INFO	ELE COURSE NUMBER	001	000
	ELE PROFESSOR NAME	001	000
	ELE SECTION	015	000
	ELE CLASS SIZE	001	000
COURSE_INFO	REC COURSE INFO	001	000
	ELE SCHEDULED EXAM FLAG	001	000
EXAM SCHEDULING INPUT	REC COURSE INFO	001	001
	REC PROFESSOR INFO	001	002
	REC ROOM INFO	001	003
FINAL EXAM SCHEDULE	REC OKAY EXAM	001	000
FINAL_EXAM SCHEDULE	REC OKAY EXAM	001	000
OKAY EXAM	ELE COURSE NUMBER	001	000
	ELE PROFESSOR NAME	001	000
	ELE SECTION	015	000
	ELE FINAL EXAM PERIOD	001	000
	ELE ROOM NUMBER	001	000
PROFESSOR INFO	ELE PROFESSOR NAME	001	000
	REC TEACHING SCHEDULE	001	000
PROF_INFO	REC PROFESSOR INFO	001	000
ROOM INFO	ELE ROOM NUMBER	001	000
	ELE REFRESHER TIME	012	000
	ELE ROOM SIZE	001	000
ROOMS TABLE INFO	ELE COURSE INDICATOR	001	000
	ELE BUILDING	001	000
	ELE FLOOR	001	000
	ELE ROOM SIZE	001	000
ROOM_INFO	REC ROOM INFO	001	000
TEACHING SCHEDULE	ELE COURSE NUMBER	001	000
	ELE COURSE TIME	001	000
	ELE REFRESHER FLAG	001	000

ELEMENT NAME	ALTERNATE NAME	DEFINITION
AVAILABLE ROOM		ROOM AVAILABLE TO SCHEDULE FINAL EXAM IN DURING FINALS WEEK
BUILDING		PREFERRED BUILDING TO SCHEDULE FINAL FOR COURSE SUBJECT
CLASS SIZE		TOTAL NUMBER OF STUDENTS IN COURSE
COURSE INDICATOR		FIRST TWO CHARCTERS OF COURSE NUMBER, INDICATES CRSE SUBJECT
COURSE NUMBER		UNIQUE NUMBER IDENTIFYING COURSE, INCLUDES SEGMENT NUMBER
COURSE TIME		DAY/TIME COURSE MEETS
FINAL EXAM PERIOD		ONE OF TWELVE 2-HOUR TIME BLOCKS TO SCHEDULE A FINAL
FLOOR		FLOOR OF BUILDING TO SCHEDULE EXAM ON
PROFESSOR NAME		NAME OF PROFESSOR TEACHING COURSE
REFRESHER FLAG		INDICATES IF COURSE IS A REFRESHER COURSE
REFRESHER TIME		IF COURSE IS A REFRESHER COURSE, DAY/TIME IT IS TAUGHT
ROOM NUMBER		ROOM NUMBER IN WHICH A COURSE EXAM IS SCHEDULED
ROOM SIZE		
SCHEDULED EXAM FLAG		FLAG INDICATING EXAM WAS ABLE TO BE SCHEDULED
SECTION		UNIQUE GROUP OF STUDENTS SCHEDULED FOR COURSE TOGETHER
SEGMENT INFO		INDICATES WHETHER CLASS IS A SEGMENT OF A COURSE

LIST OF REFERENCES

1. Yourdon, Edward, *Managing the Structured Techniques - Strategies for Software Development in the 1990s (Third Edition)*, Yourdon Press, New York, New York, 1986.

2. Davis, William S., *Systems Analysis and Design: A Structured Approach* Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

3. Page-Jones, Meiler, *The Practical Guide to Structured Systems Design*, Yourdon Press, Englewood Cliffs, New Jersey, 1980.

4. *DesignAid Tutorial*, Release 3.55, Nastec Corporation, Southfield, Michigan, 1986.

5. *DesignAid User's Guide*, Release 3.55, Nastec Corporation, Southfield, Michigan, 1986.

6. *DesignAid Reference Manual*, Release 3.55, Nastec Corporation, Southfield, Michigan, 1986.

7. *Excelerator Tutorial*, Index Technology Corporation, 1986, Cambridge, Massachusetts.

8. *Excelerator User's Guide*, Index Technology Corporation, Cambridge, Massachusetts, 1986.

9. *Excelerator Reference Guide*, Index Technology Corporation, Cambridge, Massachusetts, 1986.

10. Secretary of the Navy Instruction 5231.1B, *Life Cycle Management (LCM) Policy and Approval Requirements for Information System (IS) Projects*, 8 March 1985.

11. Marine Corps Order P5231.1A, *Life Cycle Management for Information Systems Projects (Short Title: LCM-IS) (DRAFT)*, 20 April 1987.

12. Fiegas, Dietmar, *The Naval Postgraduate School Scheduling System: A Heuristic Approach*, Master Thesis, Naval Postgraduate School, Monterey, California, September, 1985. thesis by former NPS student Dietmar W. Fiegas, September 1985.

13. Interview with Ms. Mary Horn, NPS Course Scheduler, July/Aug 1987.

14. Nastec Corporation's Toll-Free Hotline, Customer Service Representative.

15. Index Technology Corporation's Toll-Free Hotline, Customer Service Representative.

BIBLIOGRAPHY

Adams, David R., and Hetzel, William, *Computer Information Systems Development: Principles and Case Study* South-Western Publishing Company, 1985.

Department of Defense Directive 7920.1A, Life Cycle Management of Automated Information Systems (AIS), dated 17 October 1978.

United States Marine Corps Information Resources Management Standards and Guidelines Program Technical Publication IRM-5231-20 (Requirements Statement), dated 26 February 1987.

United States Marine Corps System Development Methodology, Volume III, Standards, Supplement Number 10 (Functional Requirements Definition Standard), draft dated February 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-95) Navy Department Washington, D. C. 20350 - 2000	1
4.	Barry A. Frew Faculty Code 54Fw U. S. Naval Postgraduate School Monterey, CA 93943	1
5.	Professor Nancy Roberts Faculty Code 54Rc U. S. Naval Postgraduate School Monterey, CA 93943	1
6.	Commandant of the Marine Corps (Code MPI-40) (ATTN: LtCol AXTELL) Washington, D. C. 20380 - 0001	2
7.	Marine Corps Finance Center (Code RFAM-KC) (ATTN: Maj GRABO) Kansas City, Missouri 64197 - 0001	2
8.	Fleet Numerical Oceanographic Center (Code 008) Monterey, California 93943	1
9.	LCDR T. Dove COMNAVINTCOM (00J) 4600 Silver Hill Road Washington, D. C. 20389 - 5000	1
10.	Ms. Mary Horn Scheduler, U. S. Naval Postgraduate School	2
11.	Captain D. A. Ganzer Headquarters, U. S. Marine Corps (Code CCIS) Washington, D. C. 20380	2

Thesis
G175
c.1

Ganzer

Using Computer-Aided Software Engineering (CASE) tools to document the current logical model of a system for DoD requirements sprcifications.